

NAVAL POSTGRADUATE SCHOOL Monterey, California

AD-A223 742



THESIS

DTIC
JUL 13 1990

A REDUCED-ORDER EXTENDED KALMAN
FILTER FOR MOVING IMAGES

by

Philip A. Lindeman

December 1989

Thesis Advisor:

Jeffrey B. Burl

Approved for public release; distribution is unlimited

90 07 11 132

Approved for public release; distribution is unlimited

A Reduced-Order Kalman Filter for
Moving Images

by

Philip A. Lindeman
Captain, United States Marine Corps
B.S., Biology, West Virginia Wesleyan College, 1980

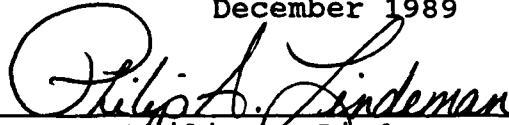
Submitted in partial fulfillment
of the requirements for the degree of

MASTER of SCIENCE in ELECTRICAL ENGINEERING

from the

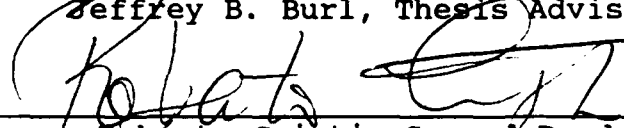
NAVAL POSTGRADUATE SCHOOL
December 1989

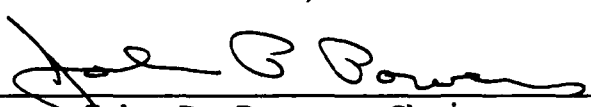
Author:


Philip A. Lindeman

Approved by:

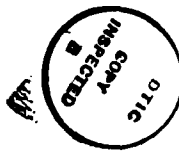

Jeffrey B. Burl, Thesis Advisor


Roberto Cristi, Second Reader


John P. Powers, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

An extended Kalman filter is used to estimate the velocity of an object moving across an image frame and to reduce the undesirable effects of noise. The extended Kalman filter is implemented in the spatial frequency domain to reduce the number of computations. The resulting filter structure is a parallel bank of third-order extended Kalman filters. This parallel structure is referred to as the modified extended Kalman filter. The performance of the modified extended Kalman filter is evaluated under a variety of noise conditions using computer simulations. Simulations employed two test objects moving across a reference image in the presence of zero-mean, white, Gaussian noise. The performance of the filter was demonstrated when these objects were moved at integer and noninteger velocities. Performance was also evaluated when a stationary background was included with the white noise.



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist _____	
A-1	

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made within the time available to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	THE MOVING IMAGE MODEL.....	3
	A. THE MOVING IMAGE MODEL IN THE SPATIAL DOMAIN.....	3
	B. DERIVATION OF THE SHIFT OPERATOR IN THE SPATIAL FREQUENCY DOMAIN.....	4
	C. THE MOVING IMAGE MODEL IN THE SPATIAL FREQUENCY DOMAIN.....	7
III.	THE EXTENDED KALMAN FILTER.....	11
	A. GENERAL.....	11
	B. THE EXTENDED KALMAN FILTER.....	11
	C. MODELING PROBLEMS.....	13
	D. THE MODIFIED EXTENDED KALMAN FILTER.....	14
	E. THE REDUCED-ORDER EXTENDED KALMAN FILTER.....	17
IV.	SIMULATIONS.....	19
	A. PERIODICITY OF THE TEST OBJECTS.....	19
	B. DISCRETE FOURIER TRANSFORM OF MOVING IMAGES.....	22
	C. LOWPASS FILTERING OF THE TRANSFORMED IMAGES.....	25
	D. PERFORMANCE MEASURES OF THE EXTENDED KALMAN FILTER.....	27
	E. SIMULATION RESULTS.....	28
V.	CONCLUSIONS.....	69
APPENDIX A TUTORIAL ON TWO-DIMENSIONAL SPATIAL FREQUENCIES.....		71
	A. PROGRAM DISCUSSION.....	71
	B. PROGRAM LISTING.....	79

APPENDIX B	PPHSE.M.....	82
A.	PROGRAM DISCUSSION.....	82
B.	PROGRAM LISTING.....	87
APPENDIX C	PMEKF.M.....	90
LIST OF REFERENCES.....		97
BIBLIOGRAPHY.....		98
INITIAL DISTRIBUTION LIST.....		99

ACKNOWLEDGEMENTS

I would like to thank Professor J. B. Burl for his patience and perseverance in assisting me with this expansive concept. I am also grateful to Professor Hal Titus and his able assistant, Colin Cooper, who provided assistance, encouragement, and tolerated my round-the-clock presence in the Computer Controls Lab. I would also like to thank two of my fellow classmates, Captain Stephen L. Spehn, USMC, and Major David Aviv, Israeli Air Force, for their encouragement and assistance. Finally, I would like to share this accomplishment with my stalwart wife, Lorna, and son, Andrew, who provided continual encouragement and tolerance towards my completing this thesis.

I. INTRODUCTION

In many radar and remote sensing operations, successive image frames may contain a moving object of interest. Identification of this object may be complicated by image degradation resulting from the presence of background noise. The idea of using an extended Kalman filter (EKF) to enhance the quality of the image frame and provide an estimate of the object velocity was proposed by Burl [Ref. 1].

In this thesis, a moving image model is defined as a series of image frames containing a moving object. An image frame consists of a two-dimensional array where the individual array elements are defined as pixels. Each pixel is given a numerical value based on the intensity of the image at that point. These values are either stored in a computer or projected on a video display terminal.

The EKF requires a model that describes the evolution of the image frames in time. This spatial domain model is derived in Chapter II. A shift operator in the model describes the motion of the object as it traverses the image frame and is a function of the object's velocity. A prohibitive number of calculations would be necessary if the EKF were implemented in the spatial domain. [Ref. 1]

Chapter II further describes the moving image model after it has been transformed to the spatial frequency domain. The

two-dimensional discrete Fourier transform (DFT) transforms these two-dimensional image frames from the spatial domain to the spatial frequency domain. By transforming the image frame to the spatial frequency domain, the EKF can analyze each of the spatial frequencies separately. The moving object is evidenced by a phase shift in each of the spatial frequencies. The implementation of the EKF in the spatial frequency domain reduces the number of calculations. The number of calculations can be further reduced by limiting the number of spatial frequencies that are analyzed. [Ref. 1]

Chapter III presents the extended Kalman filter equations. As the velocity estimate error approaches zero, the EKF is shown to converge to a parallel set of third-order extended Kalman filters. This parallel structure is referred to as the modified extended Kalman filter (MEKF). [Ref. 1]

Chapter IV discusses the results of the computer simulations that were run to record the performance of the EKF under a variety of noise conditions and background environments. Conclusions are presented in Chapter V.

II. THE MOVING IMAGE MODEL

A. THE MOVING IMAGE MODEL IN THE SPATIAL DOMAIN

Successive image frames containing a moving object can be modeled by a nonlinear state equation in the spatial domain as,

$$\begin{aligned} \mathbf{x}(k+1) &= \begin{bmatrix} \mathbf{f}(k+1) \\ \mathbf{v}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{S}(\mathbf{v}) & 0 \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}(k) \\ \mathbf{v}(k) \end{bmatrix} + \begin{bmatrix} \zeta_f(k) \\ \zeta_v(k) \end{bmatrix} \\ &= \mathbf{a}(\mathbf{v})\mathbf{x}(k) + \zeta(k) . \end{aligned} \quad (2.1)$$

An image frame has the dimensions of N_1 by N_2 pixels. The amplitude of each pixel, when stacked into a column, forms the vector \mathbf{f} , where $\mathbf{f} \in R^{(n_1 n_2)}$. The velocity vector, \mathbf{v} , consists of a two-element column vector describing the horizontal and vertical velocities of the moving object in terms of the number of pixels per iteration. The vectors \mathbf{f} and \mathbf{v} combine to form the state vector \mathbf{x} , where $\mathbf{x} \in R^{(n_1 n_2 + 2)}$. $\mathbf{S}(\mathbf{v})$ is a two-dimensional shift operator with the magnitude and direction being a function of the velocity vector, \mathbf{v} . The plant noise vector, ζ , is composed of the image plant noise, ζ_f , and the velocity plant noise, ζ_v . The plant noise is assumed to be a zero-mean, white, Gaussian random process with a covariance matrix defined as

$$Q_{\zeta\zeta} = E \left[\begin{bmatrix} \zeta_f \\ \zeta_v \end{bmatrix} \begin{bmatrix} \zeta_f & \zeta_v \end{bmatrix} \right] = \begin{bmatrix} \sigma_f^2 I & 0 \\ 0 & \sigma_v^2 I \end{bmatrix}, \quad (2.2)$$

where $E[\cdot]$ is the expectation operator and I is the identity operator.

The measured image, $y(k)$, is defined by the corresponding measurement equation,

$$\begin{aligned} y(k) &= f(k) + v(k) \\ &= \begin{bmatrix} I & 0 \end{bmatrix} x(k) + v(k) \\ &= c x(k) + v(k), \end{aligned} \quad (2.3)$$

where $y \in R^{(n_1 n_2)}$ and $v \in R^{(n_1 n_2)}$. The zero mean, white, Gaussian random process, $v(k)$, has a known covariance matrix

$$R_{vv} = E [v(k) v^T(k)] = \sigma_v^2 I. \quad (2.4)$$

B. DERIVATION OF THE SHIFT OPERATOR IN THE SPATIAL FREQUENCY DOMAIN

A image frame is a two-dimensional, finite-extent sequence defined as $f(n_1, n_2)$ where $0 \leq n_1 \leq N_1-1$ and $0 \leq n_2 \leq N_2-1$. The sequence $f(n_1, n_2)$ is transformed from the spatial domain to the spatial frequency domain using the two-dimensional discrete Fourier transform (DFT),

$$F(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1, n_2) \exp \left[-j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right) \right], \quad (2.5)$$

where $0 \leq k_1 \leq N_1-1$ and $0 \leq k_2 \leq N_2-1$.

The shift operator in the spatial frequency domain is derived from the circular shift property of the discrete Fourier transform [Ref. 2:p. 67]. The circular shift property is defined as

$$\begin{aligned} x(((n_1-m_1))_{N_1}, ((n_2-m_2))_{N_2}) &\leftrightarrow W_{N_1}^{m_1 k_1} W_{N_2}^{m_2 k_2} X(k_1, k_2) \\ &\leftrightarrow D(m_1, m_2) X(k_1, k_2), \end{aligned} \quad (2.6)$$

where $((\bullet))_{N_i}$ is the modulus operator, $W_{N_i} = e^{-j\frac{2\pi}{N_i}}$, and $D(m_1, m_2)$ is the transformed shift operator. Equation 2.6 indicates that a circular shift in the spatial domain results in a phase shift in the spatial frequency domain.

The original image frame, $x(n_1, n_2)$, is assumed to be a periodic two-dimensional sequence with the fundamental period equal to the frame dimensions. The object moving across an image frame then describes a circular shift. [Ref. 2:pp. 61-62]. This implies that, as the object moves off one edge of the screen, it reenters the screen from the opposite side. This assumption was necessary because the processing capabilities of the computer used to execute the EKF algorithm limited the image frame dimensions. Image frames of finite extent, such as a radar screen or video display, do not require this property because the moving object's size will likely be disproportionally smaller than the screen dimensions.

Derivation of $D(m_1, m_2)$ begins by introducing a circular shift into the original image frame, $f(n_1, n_2)$. Equation 2.5 now becomes

$$F'(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1-m_1, n_2-m_2) e^{-j\left(\frac{2\pi n_1 k_1}{N_1} + \frac{2\pi n_2 k_2}{N_2}\right)}, \quad (2.7)$$

where $F'(k_1, k_2)$ is the circular-shifted, transformed image frame. The amount of pixel movement is described by m_1 for the horizontal shift and m_2 for the vertical shift.

Letting $l_1 = n_1 - m_1$ and $l_2 = n_2 - m_2$, Equation 2.7 becomes

$$F'(k_1, k_2) = \sum_{l_1=-m_1}^{N_1-m_1-1} \sum_{l_2=-m_2}^{N_2-m_2-1} f(l_1, l_2) e^{-j\left(\frac{2\pi(l_1+m_1)k_1}{N_1} + \frac{2\pi(l_2+m_2)k_2}{N_2}\right)}. \quad (2.8)$$

Rearranging terms, Equation 2.8 reduces to

$$\begin{aligned} F'(k_1, k_2) &= e^{-j\left(\frac{2\pi m_1 k_1}{N_1} + \frac{2\pi m_2 k_2}{N_2}\right)} F(k_1, k_2) \\ &= W_{N_1}^{m_1 k_1} W_{N_2}^{m_2 k_2} F(k_1, k_2) \\ &= D(m_1, m_2) F(k_1, k_2), \end{aligned} \quad (2.9)$$

where $F(k_1, k_2)$ is the transformed image frame without the circular shift. This proves that, when an object moves across an image frame, the motion is described by a phase shift in the spatial frequency domain, thereby agreeing with the circular shift property (Eqn. 2.6).

The notation is simplified by defining the frequency vector, ω , as

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} \frac{2\pi k_1}{N_1} \\ \frac{2\pi k_2}{N_2} \end{bmatrix} \quad (2.10)$$

and the velocity vector as

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} . \quad (2.11)$$

Equation 2.9 now becomes

$$\begin{aligned} F'(k_1, k_2) &= e^{-j(\omega^T v)} F(k_1, k_2) \\ &= D(m_1, m_2) F(k_1, k_2) \\ &= D(v) F(k_1, k_2), \end{aligned} \quad (2.12)$$

where T is the transpose operation.

The example contained in Appendix A further demonstrates the use of the phase shift operator in reconstructing image frames in the spatial frequency domain.

C. THE MOVING IMAGE MODEL IN THE SPATIAL FREQUENCY DOMAIN

The spatial domain model described by Equations 2.1 and 2.3 can now be transformed to the spatial frequency domain. The transformation is accomplished by using the two-dimensional discrete Fourier transform,

$$F(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1, n_2) \exp \left[-j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right) \right] , \quad (2.13)$$

where $0 \leq k_1 \leq N_1-1$ and $0 \leq k_2 \leq N_2-1$.

The vector, $f(k)$, and its corresponding plant noise, ζ_f , are transformed to the spatial frequency domain,

$$F(k) = F\{f(k)\} = \begin{bmatrix} \text{Re}(F_1(k)) \\ \text{Im}(F_1(k)) \\ \vdots \\ \text{Re}\left(\frac{F_{N_1 N_2}(k)}{2}\right) \\ \text{Im}\left(\frac{F_{N_1 N_2}(k)}{2}\right) \\ v(k) \end{bmatrix} \quad (2.14)$$

$$Z(k) = F\{\zeta_f(k)\} , \quad (2.15)$$

where $F(\bullet)$ is the two-dimensional discrete Fourier transform. (It should be noted that the index k refers to the sample at time k and is not related to the frequency vector.) Only half of the spatial frequencies need to be stored in $F(k)$ due to the conjugate symmetry that results from the transformation of a real image [Ref. 3:p. 45].

The state equation is defined in the spatial frequency domain as,

$$X(k+1) = A(v)X(k) + Z(k)$$

$$\begin{bmatrix} F(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} D(v) & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} F(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} z_f(k) \\ \zeta_v(k) \end{bmatrix} , \quad (2.16)$$

where $X \in R^{(n_1 n_2 + 2)}$. The transformed shift operator, $D(v)$, is a diagonal matrix where the diagonal terms, $D_i(v)$, are the

transformed shift operators for a specific spatial frequency. Because the real and imaginary parts of the spatial frequency vector are stacked into a column vector, $D_i(v)$ is expressed in terms of its trigonometric relations,

$$F_i(k+1) = D_i(v)F_i(k)$$

$$\begin{bmatrix} \text{Re}(F_i(k+1)) \\ \text{Im}(F_i(k+1)) \end{bmatrix} = \begin{bmatrix} \cos(\omega_i^T v) & \sin(\omega_i^T v) \\ -\sin(\omega_i^T v) & \cos(\omega_i^T v) \end{bmatrix} \begin{bmatrix} \text{Re}(F_i(k)) \\ \text{Im}(F_i(k)) \end{bmatrix}, \quad (2.17)$$

where $i = 0, 1, \dots, \frac{(N_1-1)(N_2-1)}{2}$.

The covariance matrix for the image plant noise, Z , is,

$$Q_{\zeta\zeta} = E [Z Z^T] = \begin{bmatrix} \sigma_f^2 I & 0 \\ 0 & \sigma_v^2 I \end{bmatrix}, \quad (2.18)$$

where $E[\cdot]$ is the expectation operator and I is the identity operator. Because the discrete Fourier transform is a linear operation on ζ_f , Z remains a zero-mean, Gaussian random process.

The corresponding measurement equation in the spatial frequency domain becomes,

$$\begin{aligned} Y(k) &= F(k) + N(k) \\ &= [I \ 0] X(k) + N(k) \\ &= C X(k) + N(k) \end{aligned} \quad (2.19)$$

where $Y(k) = F\{y(k)\}$ and $N(k) = F\{\nu(k)\}$. The state equations given by Equations 2.16 and 2.19 are the basis for implementing the EKF in the spatial frequency domain.

III. THE EXTENDED KALMAN FILTER

A. GENERAL

In this chapter, the extended Kalman filter (EKF) is obtained from the state equations that were derived for the moving image model. Equations 2.16 and 2.19 are used to implement the EKF in the spatial frequency domain. The modified extended Kalman filter (MEKF) is based on the diagonal properties of the transformed shift operator, $D(v)$, and covariance matrices. The parallel structure of the MEKF is formed in the limit as the velocity estimate approaches the actual velocity.

B. THE EXTENDED KALMAN FILTER

The extended Kalman filter (EKF) algorithm is used when the state equations are linearized about each new estimate as they become available [Ref. 4: p.158]. The EKF uses the spatial frequency domain state equations, Eqns 2.16 and 2.19, for the moving image model.

The linearized state equation is given as

$$\delta X(k+1) = \hat{A}(X_0) \delta X(k) + Z(k) , \quad (3.1)$$

where $\delta X(k) = X(k) - X_0$. Here, X_0 is shorthand notation for the current estimate $\hat{X}(k|k)$ [Ref. 4: p.158] and $\hat{A}(X_0)$ is the Jacobian of the nonlinear state equations about X_0 . This

means that the ij component of $\hat{A}(X_0)$ is the partial derivative with respect to the j th state of the i th component of the state equation (Eqn. 2.16) and $\hat{A}(X_0)$ is then composed as follows:

$$\hat{A}(X_0) = \text{diagonal}[\mathbf{A}_i(X_0)] \quad i = 1, 2, \dots, \frac{N_1 N_2}{2} \quad (3.2)$$

$$\hat{A}_i(X_0) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & [-f_1(k)\sin(\alpha) + f_2(k)\cos(\alpha)]\omega^T_i \\ -\sin(\alpha) & \cos(\alpha) & [-f_1(k)\cos(\alpha) - f_2(k)\sin(\alpha)]\omega^T_i \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.3)$$

where $\alpha = \omega^T_i v$, $f_1(k) = \text{Re}[F_i(k)]$ and $f_2(k) = \text{Im}[F_i(k)]$. A further discussion of the linearization techniques used by EKFs is contained in Ref. 4, p.154, and Ref. 5, p.194.

The EKF equations for the moving image model are summarized below.

Prediction

State prediction

$$\hat{X}(k+1|k) = A(v) \hat{X}(k|k) \quad (3.4)$$

Covariance prediction

$$P(k+1|k) = \hat{A}(X_0) \hat{X}(k|k) P(k|k) \hat{A}^T(X_0) \hat{X}(k|k) + Q_{zz} \quad (3.5)$$

Innovation

$$\begin{aligned} e(k+1) &= Y(k+1) - \hat{Y}(k+1|k) \\ &= Y(k+1) - C \hat{X}(k+1|k) \end{aligned} \quad (3.6)$$

Kalman gain

$$G(k+1) = P(k+1|k) C^T [C P(k+1|k) C^T + R_{NN}]^{-1} \quad (3.7)$$

Correction

State correction

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + G(k+1) [Y(k+1) - \hat{Y}(k+1|k)] \quad (3.8)$$

Covariance correction

$$P(k+1|k+1) = [I - G(k+1)C]P(k+1|k) \quad (3.9)$$

The notation $k+1|k$ reads as "at time $k+1$, given data up through time k ."

B. MODELING PROBLEMS

When the Kalman filter is implemented, the operation of the filter can be degraded by modeling errors resulting from linearization around the imperfectly estimated state. One possible method of reducing this divergence is to add fictitious plant noise to the system model. [Ref. 6: p.280]

The system dynamics matrix, $A(v_0 + \Delta)$, is assumed to have model errors which is represented by Δ in Eqn. 3.10 and is expressed as a first-order Taylor series expansion. The first-order term is then combined with the actual plant noise vector, $w(k)$, to form the plant noise vector, ζ .

$$\begin{aligned} X(k+1) &= A(v_0 + \Delta)X(k) + w(k) \\ &\cong A(v)X(k) + \left[\frac{\partial A(v)}{\partial v} \bigg|_{v=v_0} \Delta X(k) + w(k) \right] \\ &= A(v)X(k) + \zeta(k) , \end{aligned} \quad (3.10)$$

where Δ denotes the model error and $w(k)$ is the actual plant noise vector. Including the model error, as part of the plant

noise vector, prevents new measurements from being weighted too lightly because the Kalman gains tend to zero as $t \rightarrow \infty$. This ensures that the model continues to track the system as new measurements are taken.

C. THE MODIFIED EXTENDED KALMAN FILTER

As mentioned earlier, the diagonal properties of the transformed shift operator, $D(v)$, and the covariance matrices are the basis for the parallel structure shown in Figure 3.1.

This parallel structure is referred to as the modified extended Kalman filter. Reference 1 outlines how the EKF converges to the MEKF as the velocity estimate approaches the actual velocity.

Practical implementation of the MEKF suggests that each filter in the parallel bank be dedicated to a specific spatial frequency. Each individual filter is referred to as a single frequency extended Kalman filter (SFEKF).

The state vector for a specific spatial frequency is defined as:

$$X_i(k) = \begin{bmatrix} X_{i,1}(k) \\ X_{i,2}(k) \\ X_{i,3}(k) \end{bmatrix} = \begin{bmatrix} \text{Re}(F_i(k)) \\ \text{Im}(F_i(k)) \\ \omega_i^T v \end{bmatrix}, \quad (3.11)$$

where the first two states, $X_{i,1}(k)$ and $X_{i,2}(k)$, are the Fourier coefficients associated with a specific spatial frequency and the third state, $X_{i,3}(k)$, is the dot product of the spatial

frequency vector and the velocity vector. The reason for making $X_{i,3}(k)$ a dot product is because the velocity vector is not directly observable for a specific spatial frequency. Even though the Kalman filter can generate estimates without the state being observable, estimation of the unobservable portion of the state relies on a priori information and not on the measurements [Ref. 1]. Simplification of the individual filters results because only $X_{i,3}(k)$ is estimated. The a priori information is later combined with all the estimates of $X_{i,3}(k)$ to yield a final velocity estimate. The final velocity estimate is calculated using the weighted least squares algorithm given by:

$$\hat{v}(k) = (\omega^T \Sigma^{-1} \omega)^{-1} \omega^T \Sigma^{-1} \hat{X}_{all,3}(k) , \quad (3.12)$$

where ω was defined by Eqn. 2.10 and

$$\Sigma^{-1} = \begin{bmatrix} \frac{1}{P_{i,3,3}(k|k)} & 0 \\ \vdots & \vdots \\ 0 & \frac{1}{P_{\frac{N_1 N_2}{2},3,3}(k|k)} \end{bmatrix} . \quad (3.13)$$

$P_{i,3,3}$ is the 3,3 component of the i th estimation error covariance matrix computed by each of the SFEKF's.

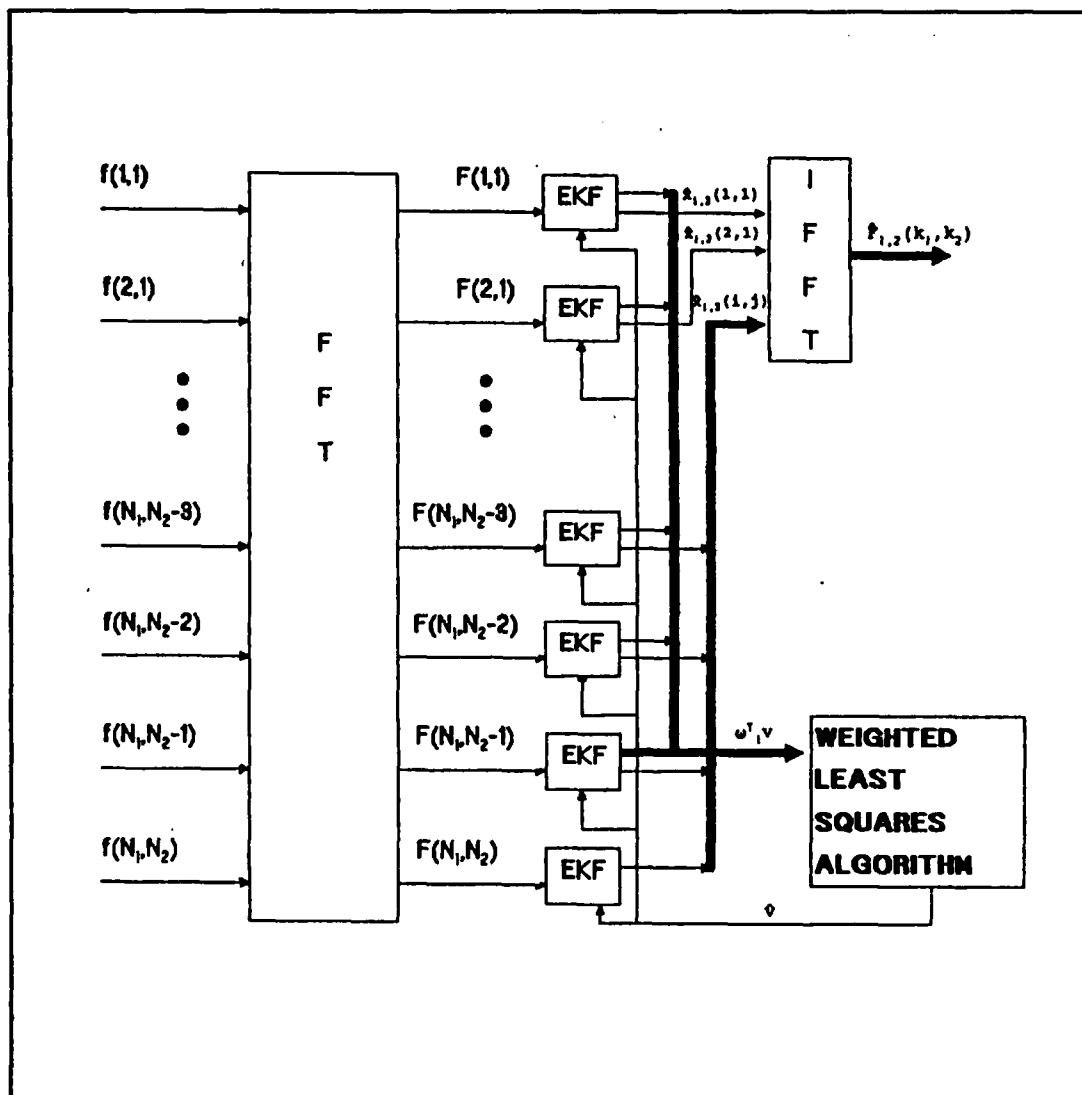


Figure 3.1 The Modified Extended Kalman Filter

The state equation for the SFEKF is

$$\begin{aligned} X_i(k+1) &= A_i(v)X_i(k) + Z_i(k) \\ &= \begin{bmatrix} D_i(v) & 0 \\ 0 & 1 \end{bmatrix} X_i(k) + Z_i(k) \end{aligned} \quad (3.14)$$

and Z_i has the known covariance matrix

$$Q_{Z_i Z_i} = E[Z_i Z_i^T] = \begin{bmatrix} \sigma_f^2 & 0 & 0 \\ 0 & \sigma_f^2 & 0 \\ 0 & 0 & \|\omega_i\|^2 \sigma_v^2 \end{bmatrix}. \quad (3.15)$$

The measurement equation and corresponding measurement noise covariance matrix for the SFEKF are similarly derived for each spatial frequency and have the form shown earlier in Eqn. 2.19.

The linearized state equation is given by

$$\delta X_i(k+1) = \hat{A}_i(X_{0_i}) \delta X_i(k) + Z_i(k), \quad (3.16)$$

where $\hat{A}_i(X_{0_i})$ is the Jacobian for a specific spatial frequency and is linearized about X_{0_i} .

The extended Kalman filter equations, Eqns. 3.3 through 3.8, are similarly modified to accommodate the SFEKF model.

D. THE REDUCED ORDER EXTENDED KALMAN FILTER

Data compression of images is achieved by truncating spatial frequencies [Ref. 1]. The parallel structure of the MEKF can easily accommodate this truncation and suggests a reduced-order filter. Even though this will lead to a

deterioration of the output image quality, the degrading effects of the image noise are simultaneously reduced.

The actual selection of which spatial frequencies to truncate will depend upon the distribution of the energy in the signal and noise level. A priori knowledge of the spatial frequency content would assist in the selection of which spatial frequencies are to be analyzed when filtering a series of image frames.

IV. SIMULATIONS

The MATLAB program written for this thesis simulated an image frame with the dimensions of 32x32 pixels and is contained in Appendix C. The two types of moving objects employed for test purposes were an 8x8 pixel square of unity amplitude (Figure. 4.1) and an 8x8 pixel pyramid with amplitude intervals of 0.25 (Figure 4.2). Additionally, tests were conducted using a fixed checkerboard background. Figures 4.3 and 4.4 depict the test objects moving across a checkerboard pattern. The individual checkers were 8x8 pixel squares with an amplitude of 0.25.

A. PERIODICITY OF THE TEST OBJECTS

An interesting challenge had to be overcome because of the small dimensions of the image frame. The circular shift property of the discrete Fourier transform allowed the computer simulation to be carried out over several hundred iterations. The original 32x32 pixel image frame used in the simulations was assumed to be a periodic two-dimensional sequence. The moving object quickly traversed the image frame because of the small dimensions. By assuming periodicity, test objects that went off one edge of the frame would reappear on the opposite edge. This assumption allowed simulations to be run over several hundred iterations because

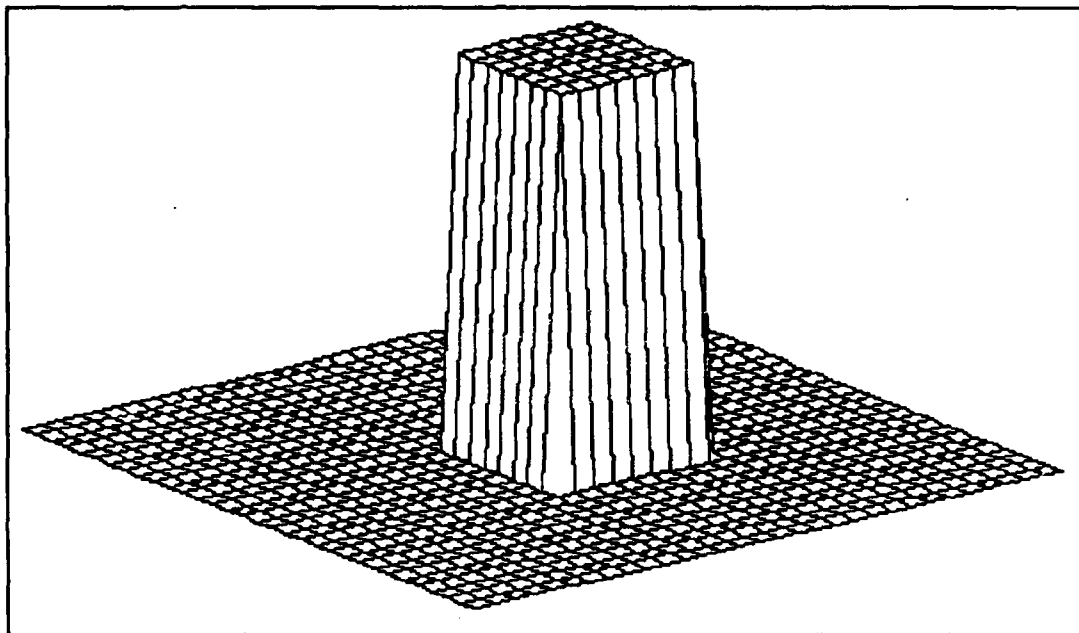


Figure 4.1 An 8x8 pixel, square object of unity amplitude.

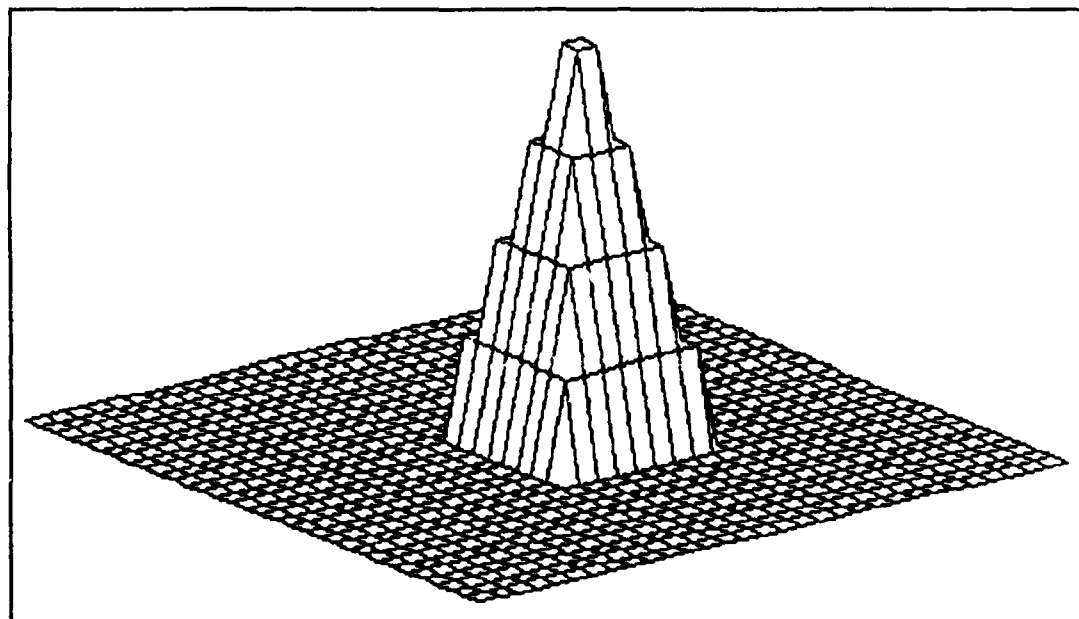


Figure 4.2 An 8x8 pixel, pyramid object with amplitude intervals of 0.25.

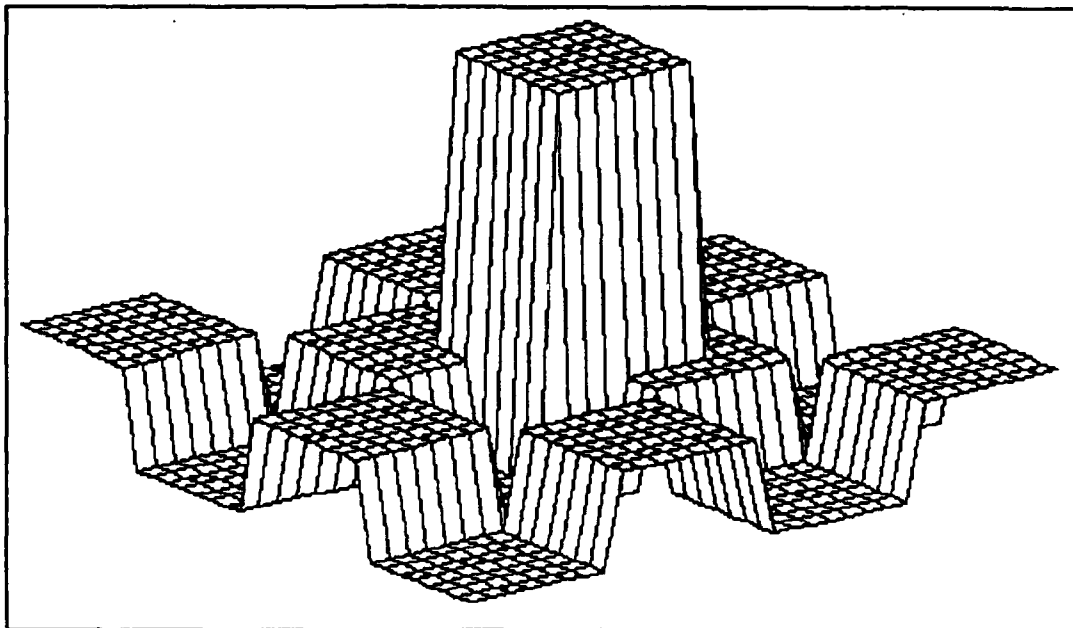


Figure 4.3 An 8x8 pixel, square object of unity amplitude moving across a checkerboard background where each of the individual checkers has an amplitude of 0.25.

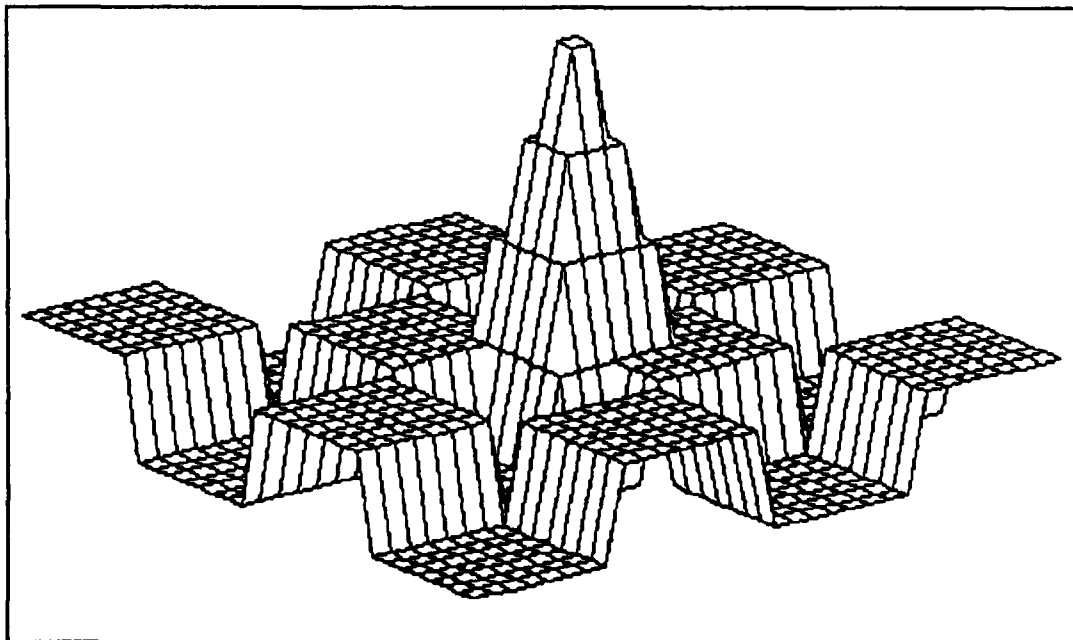


Figure 4.4 An 8x8 pixel, pyramid object with amplitude intervals of 0.25 moving across a checkerboard background where the individual checkers have an amplitude of 0.25.

it appeared to the extended Kalman filter that the image frame was of a larger dimension.

B. DISCRETE FOURIER TRANSFORM OF MOVING IMAGES

A description of how the transformed image frame appears in the spatial frequency domain will help demonstrate why the extended Kalman filter was implemented in the spatial frequency domain.

As an example, the square object depicted in Figure 4.1 is transformed to the spatial frequency domain and the resulting magnitude plot is shown in Figure 4.5. Just as a one-dimensional square wave transforms to a one-dimensional sinc function, the two-dimensional square object transforms to a two-dimensional sinc function.

What is not obvious is that as the square object is moved across the image frame, a linear phase shift results in each of the spatial frequencies; the magnitude plot remains the same. The amount of phase shift is frequency dependent as was shown in Equation 2.21.

Figure 4.6 shows the linear phase shift that occurs in the spatial frequencies $F(1,0)$ and $F(0,1)$. When the square object is moved horizontally 1 pixel per iteration, a phase shift of -11.25° occurs in $F(1,0)$. Likewise, when the square image is simultaneously shifted vertically 2 pixels per iteration, $F(0,1)$ experiences a 22.5° phase shift per iteration.

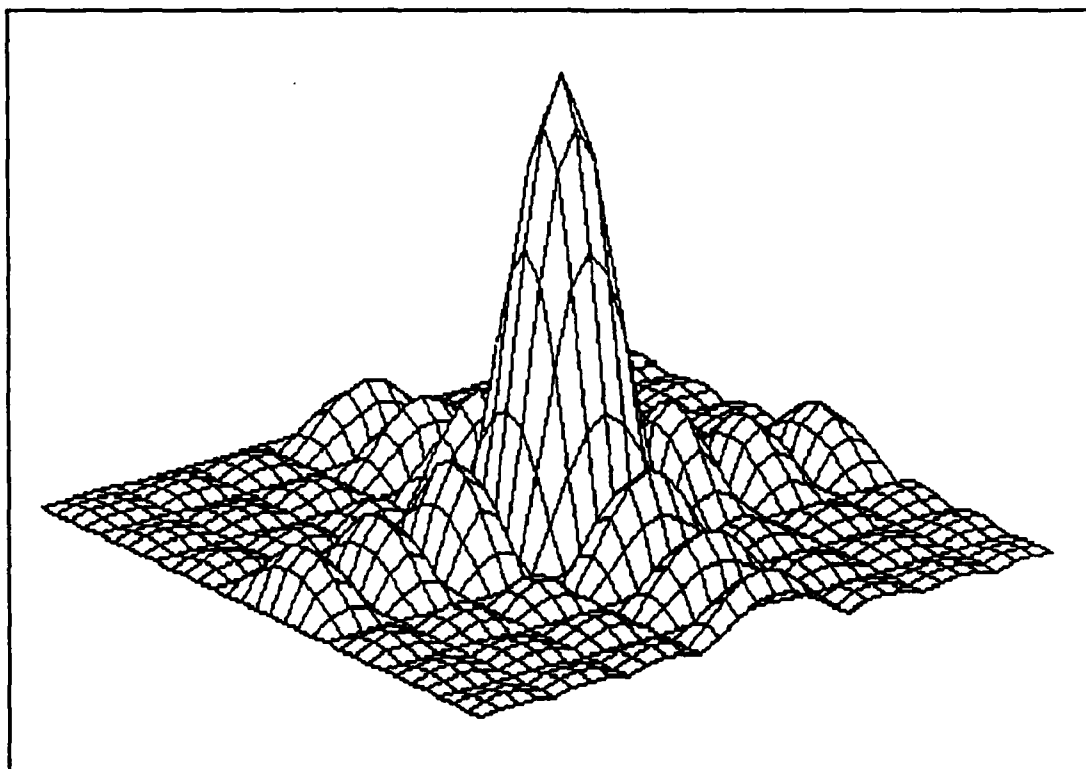


Figure 4.5 Magnitude plot of the transformed image frame depicted in Fig. 4.1.

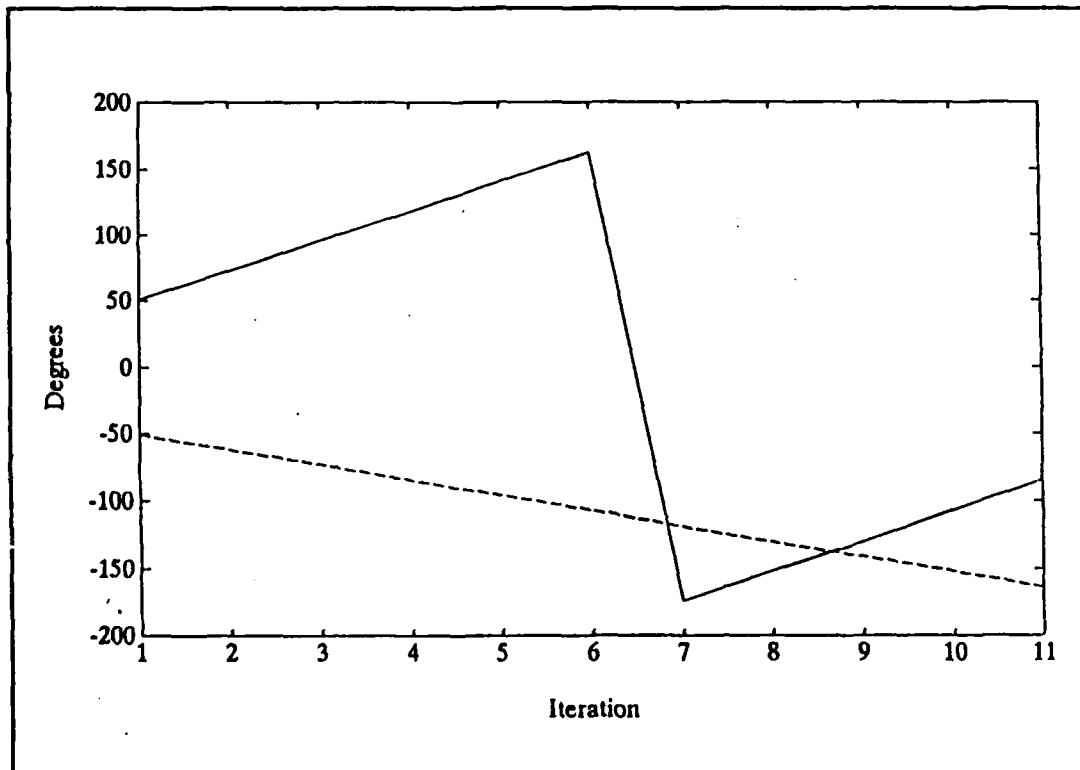


Figure 4.6 A linear phase shift of -11.25° per iteration occurs in $F(1,0)$ (dashed line) and 22.5° per iteration for $F(0,1)$ (solid line) when the square object is moved 1 pixel horizontally and 2 pixels vertically per iteration.

Appendix B further discusses the idea of how objects moving at a constant velocity in a no-noise environment create a linear phase shift in the spatial frequency domain. Also shown is the effect of adding white noise to the image frame before it is transformed to the spatial frequency domain.

C. LOWPASS FILTERING OF THE TRANSFORMED IMAGES

After the test objects were transformed to the spatial frequency domain, the specific spatial frequencies were analyzed. Most of the energy for the two test objects was centered about the dc component which was located at the center of the transformed array.

Because most of the energy was found near the dc component, the high frequency components were truncated or filtered out using a lowpass filter,

$$Y(k_1, k_2) = X(k_1, k_2)H(k_1, k_2) \quad (4.1)$$

where $Y(k_1, k_2)$ is the filtered image, $X(k_1, k_2)$ is the transformed image, and $H(k_1, k_2)$ is the lowpass filter.

As an example, Figure 4.1 was transformed to the spatial frequency domain where a lowpass filter was applied. This operation truncated the high frequency components. After transformation back to the spatial domain, Figure 4.7 shows the resulting filtered image frame. This filtered image frame

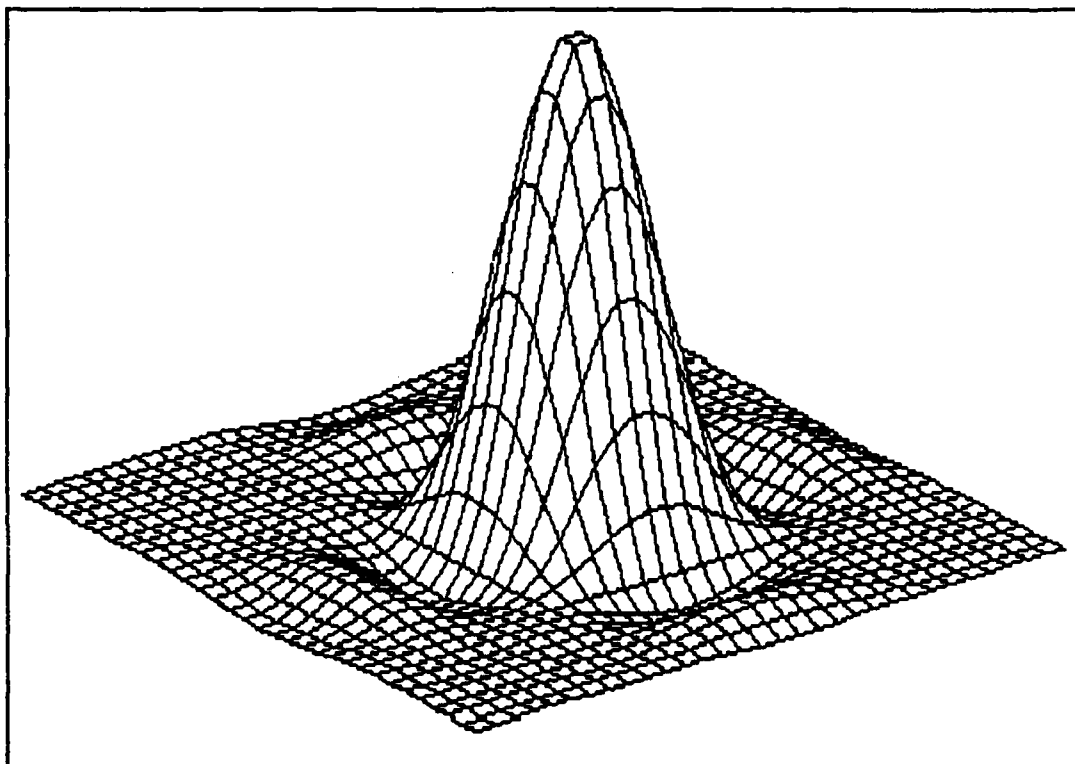


Figure 4.7 The square image depicted in Figure 4.1 after being filtered in the spatial frequency domain. The sharp edges are lost due to the loss of the high frequency components.

will be used in finding the error to calculate the Frobenius norm, which is discussed in the next section.

D. PERFORMANCE MEASURES OF THE EXTENDED KALMAN FILTER

After the specified number of iterations had been completed for each simulation, the final, filtered image frame was transformed back to the spatial domain. This filtered image frame could then be visually compared with the no-noise image frame to see how effective the extended Kalman filter performed.

An additional performance measure involved analytically calculating the Frobenius norm throughout the simulation. The Frobenius norm is a scalar which is computed by squaring each of the terms in a matrix, summing each of the squares, and then taking the square root of the final sum (Eqn. 4.1) [Ref. 7].

$$\| A \|_F = \left[\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} |a_{ij}|^2 \right]^{\frac{1}{2}} \quad (4.1)$$

Two Frobenius norms were calculated for each of the simulations. The first Frobenius norm was calculated from the error matrix that was found by subtracting the current estimate of the image frame from the current, no-noise image frame. Because the extended Kalman filter only analyzed specific spatial frequencies, the current no-noise image frame

was lowpass filtered so that only those spatial frequencies analyzed by the extended Kalman filter remained. As was shown earlier, Figure 4.1 is an example of a current, no-noise image while its lowpass filtered counterpart is depicted in Figure 4.7. The current estimate of the image frame was then subtracted from the lowpass filtered image frame and the second Frobenius norm was calculated for this error matrix.

E. SIMULATION RESULTS

The first set of computer simulations had a square test object traversing the image frame. The extended Kalman filter was evaluated under a variety of test conditions which are summarized in Table 4.1.

The two types of backgrounds were a homogeneous background with zero intensity and a checkerboard background. The velocity in Table 4.1 is described in parentheses where the first number is the horizontal velocity and the second number is the vertical velocity. Finally, the standard deviation of the white noise describes the noise level that was introduced to the image frame during the successive iterations. Additionally, to ensure that the same random noise sequence was used for each of the computer simulations, a random number seed was used to initialize the random number generator.

Figures 4.8 through 4.47 are the computer simulation results using the square test object. The no-noise image frame, that the extended Kalman filter is estimating, is

frame that the extended Kalman filter is estimating is presented first. The extended Kalman filter estimate is then depicted; followed by the horizontal and vertical velocity estimates and the Frobenius norm plots.

TABLE 4.1 SIMULATION TEST CONDITIONS (SQUARE OBJECT)

Simulation	Background	Velocity	Noise level
1	H	(1.0,2.0)	1.0
2	C	(1.0,2.0)	1.0
3	H	(1.0,2.0)	2.0
4	C	(1.0,2.0)	2.0
5	H	(1.0,2.0)	4.0
6	C	(1.0,2.0)	4.0
7	H	(2.5,1.5)	1.0
8	C	(2.5,1.5)	1.0
9	H	(1.0,0.5)	1.0
10	C	(1.0,0.5)	1.0

H - homogeneous background, C - checkerboard background

The results of the first two simulations are shown in Figures 4.8 through 4.15. The noise level for both simulations had a standard deviation of 1.0. The object was moved horizontally 1 pixel per iteration and simultaneously moved vertically 2 pixels per iteration. Even with the checkerboard background, the EKF was able to rapidly estimate the object velocity for both simulations. The Frobenius norm correspondingly reached steady state as the velocity estimates approached the actual object velocity.

The next two simulations are depicted in Figures 4.16 through 4.23. The standard deviation of the noise was increased to 2.0. Figures 4.16 and 4.20 show the square object position after 95 iterations. These two figures demonstrate the assumption that the image frame used in these simulations was a two-dimensional, periodic sequence. After 95 iterations, the square object is leaving the image frame from the far right corner. The sections of the square object that have left the image frame are shown reentering the image frame from the appropriate, opposite edges.

The EKF was able to detect the square object motion as it traversed the homogeneous background. The high noise level caused the EKF to take considerably longer to estimate the object velocity, but the EKF was still able to provide an enhance estimate of the image frame (Figure 4.17).

The increased noise level and addition of the checkerboard background precluded the EKF from detecting the square test object and estimating its velocity.

The next set of simulations depicted in Figures 4.24 through 4.31 were included to show how the EKF performed when the standard deviation of the noise was increased to 4.0. After 100 iterations, the EKF was unable to detect the square object or provide a reasonable estimate of the object velocity.

The square object velocities of simulations #7 through #10 included noninteger values. The standard deviation of

the noise was kept at 1.0 for all four simulations. The response and performance of the EKF was similar to the first two simulations where the square object moved with integer velocities and the standard deviation of the noise was 1.0.

The computer simulation results using the pyramid test object are depicted in Figures 40 through 71. The various test conditions for the pyramid object are summarized in Table 4.2.

The pyramid test object contains slightly less than half of the signal energy content in the square test object. This had a direct effect on the performance of the EKF.

Simulations #1 and #2 are shown in Figures 4.48 through 4.55. The first two simulations using the pyramid test object had identical test conditions as the first two simulations using the square test object. However, the EKF needed about 25 more iterations before it was able to track the motion of the pyramid test object (Figure 4.50).

The addition of the checkerboard background in simulation #2 further reduced the signal energy of the pyramid test object. This adversely affected the performance of the EKF as it was unable to detect or track the motion of the pyramid test object.

In simulations #3 and #4, the standard deviation of the noise was raised to 2.0. Here, the increased noise level prevented the EKF from detecting the pyramid test object.

TABLE 4.2 SIMULATION TEST CONDITIONS (PYRAMID OBJECT)

Simulation	Background	Velocity	Noise level
1	H	(1.0,2.0)	1.0
2	C	(1.0,2.0)	1.0
3	H	(1.0,2.0)	2.0
4	C	(1.0,2.0)	2.0
5	H	(2.5,1.5)	1.0
6	C	(2.5,1.5)	1.0
7	H	(1.0,0.5)	1.0
8	C	(1.0,0.5)	1.0

H - homogeneous background, C - checkerboard background

Simulations #5 through #8 had the pyramid test object moving at noninteger velocities. The standard deviation of the noise was kept at 1.0 for each of these simulations. The EKF was able to detect and closely estimate the noninteger velocities for simulations #5 and #7 which had the pyramid test object moving across the homogeneous background. Again, as was experienced in simulation #2, the addition of the checkerboard background precluded the EKF from detecting and tracking the pyramid test object.

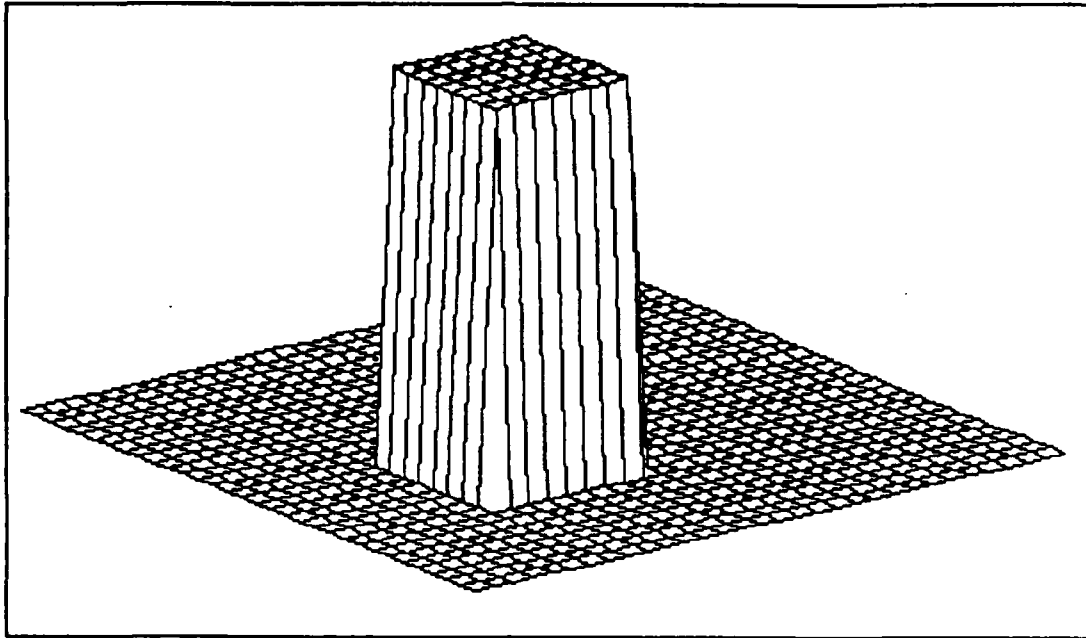


Figure 4.8 Image Frame for Simulation #1 after 40 Iterations.

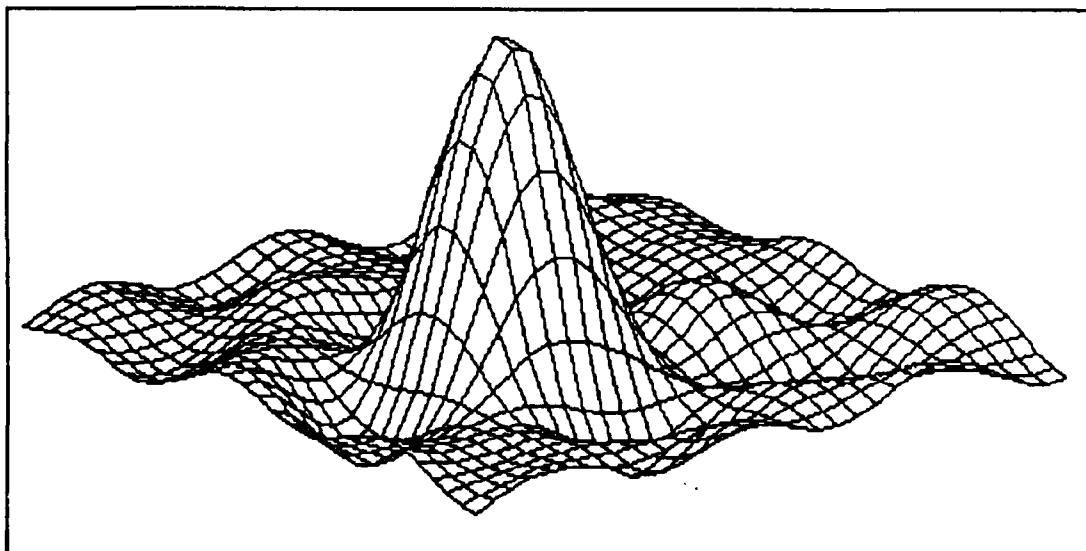


Figure 4.9 Extended Kalman Filter Estimate of Image Frame for Simulation #1 after 40 Iterations.

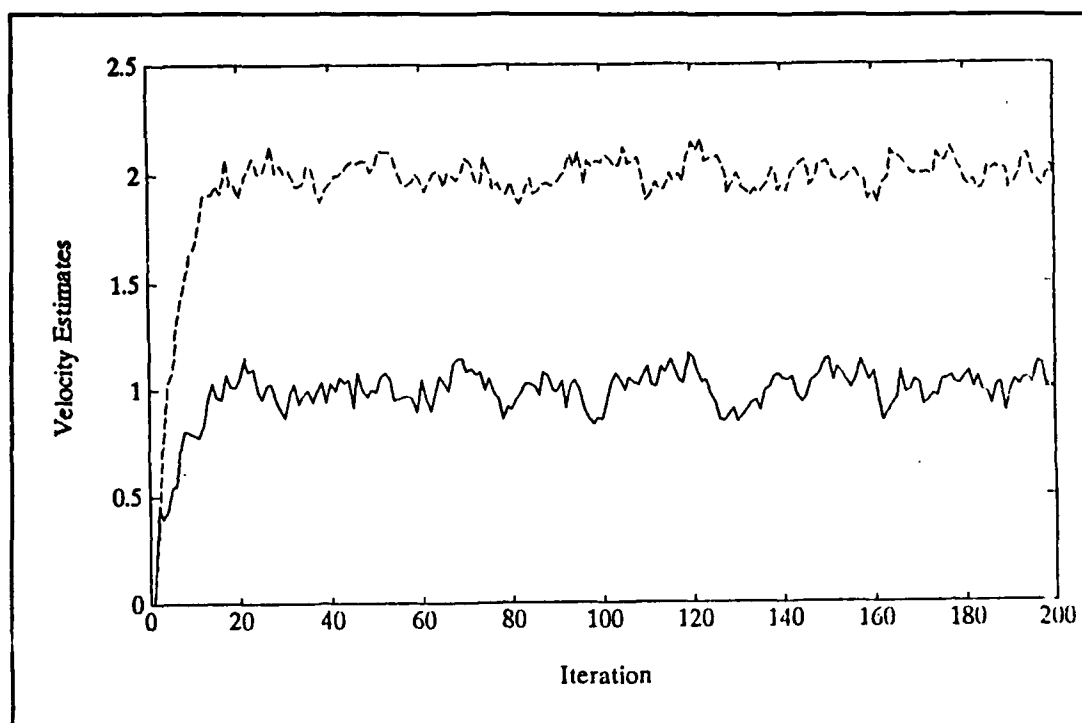


Figure 4.10 Velocity estimates for Simulation #1.

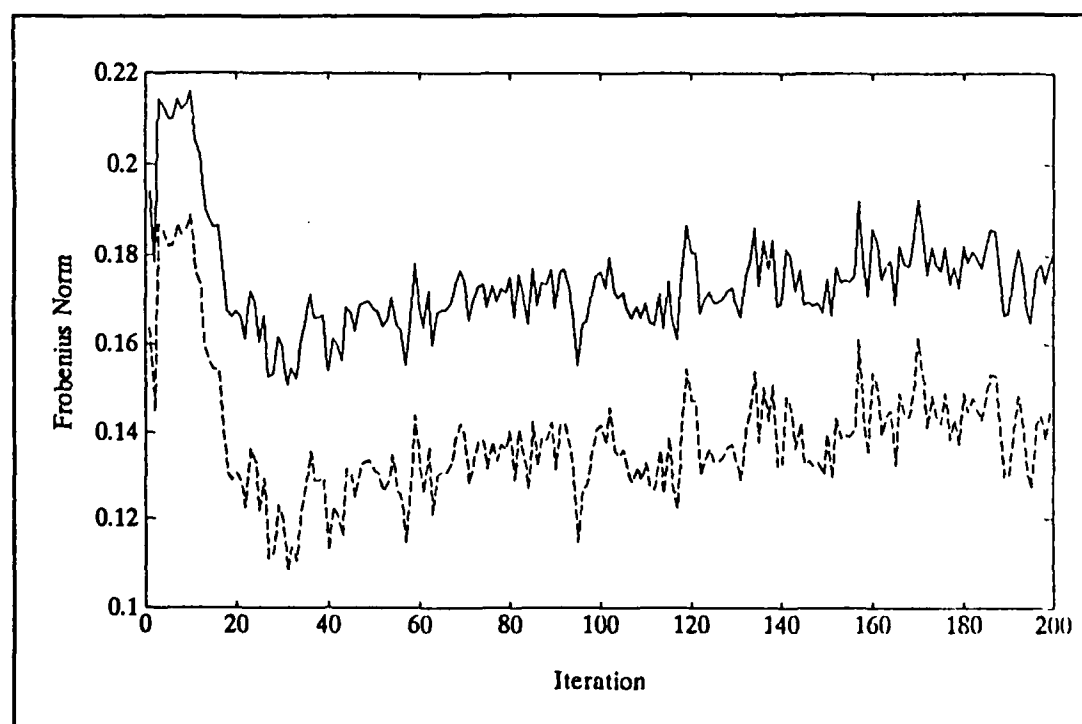


Figure 4.11 Frobenius norm results for Simulation #1.

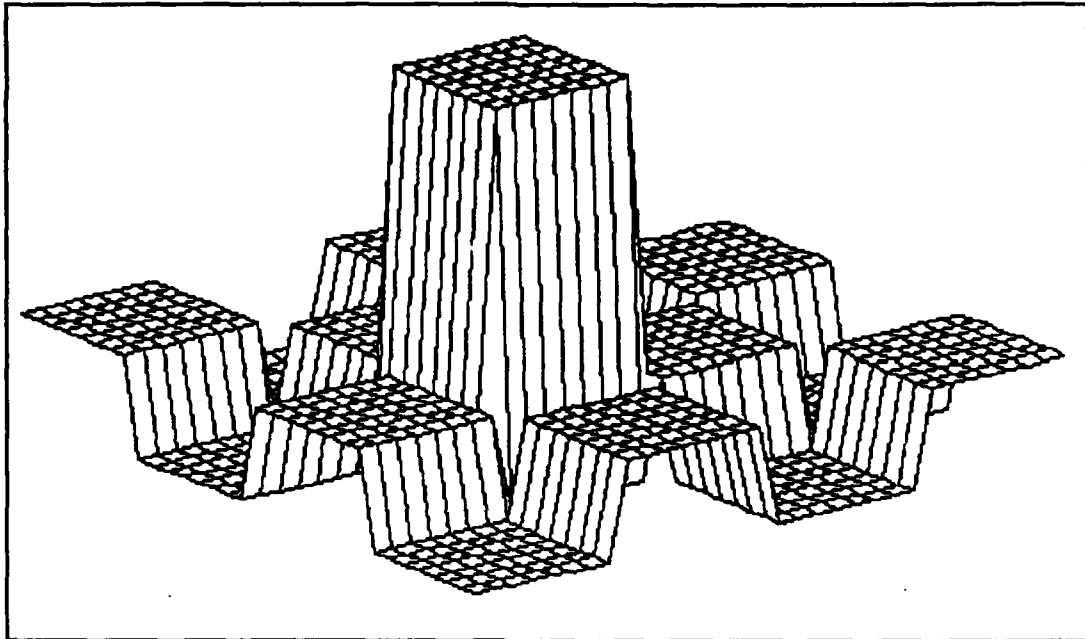


Figure 4.12 Image Frame for Simulation #2 after 40 Iterations.

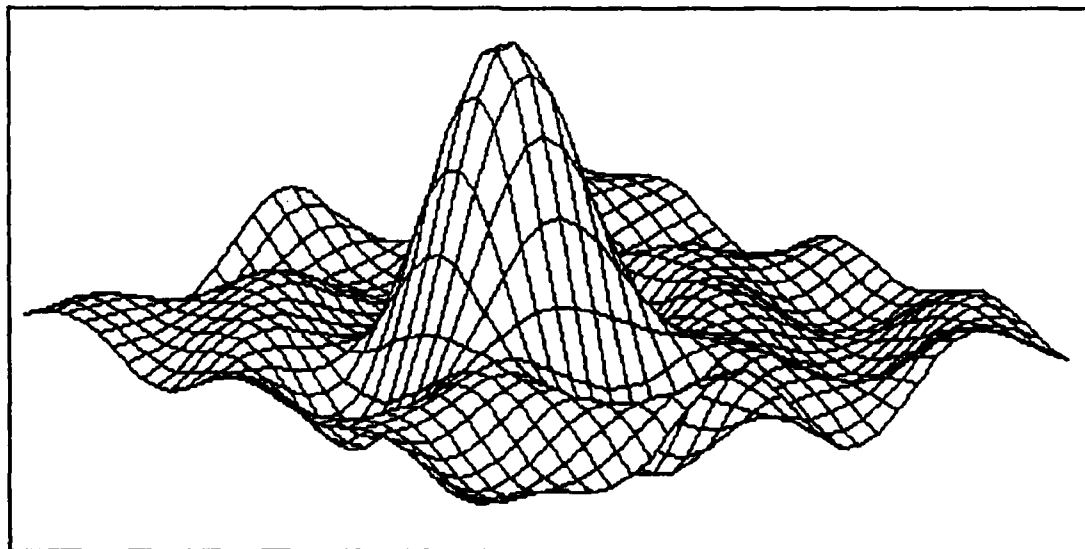


Figure 4.13 Extended Kalman Filter Estimate of Current Image Frame for Simulation #2.

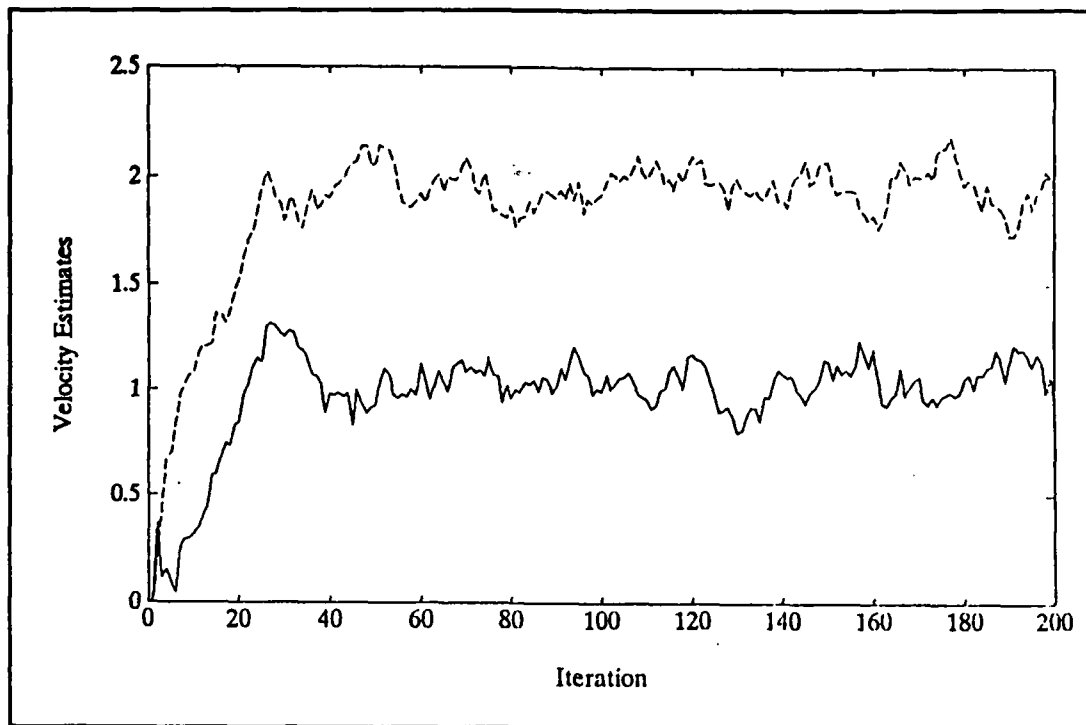


Figure 4.14 Veclocity estimates for Simulation #2.

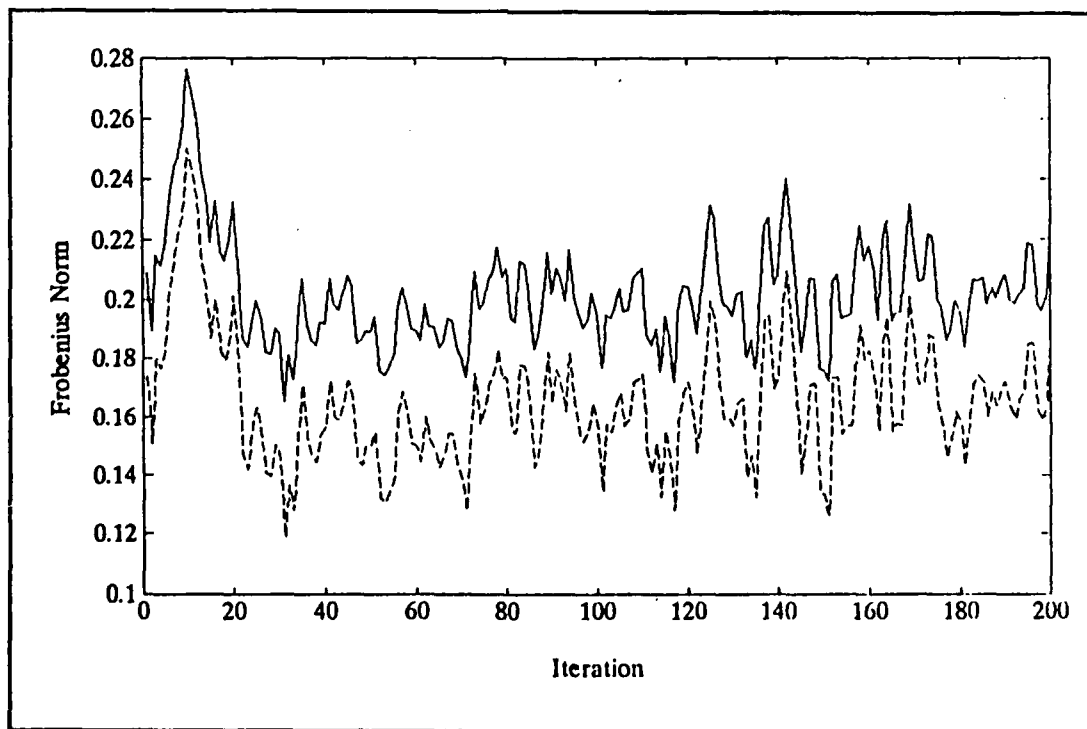


Figure 4.15 Frobenius norm results for Simulation #2.

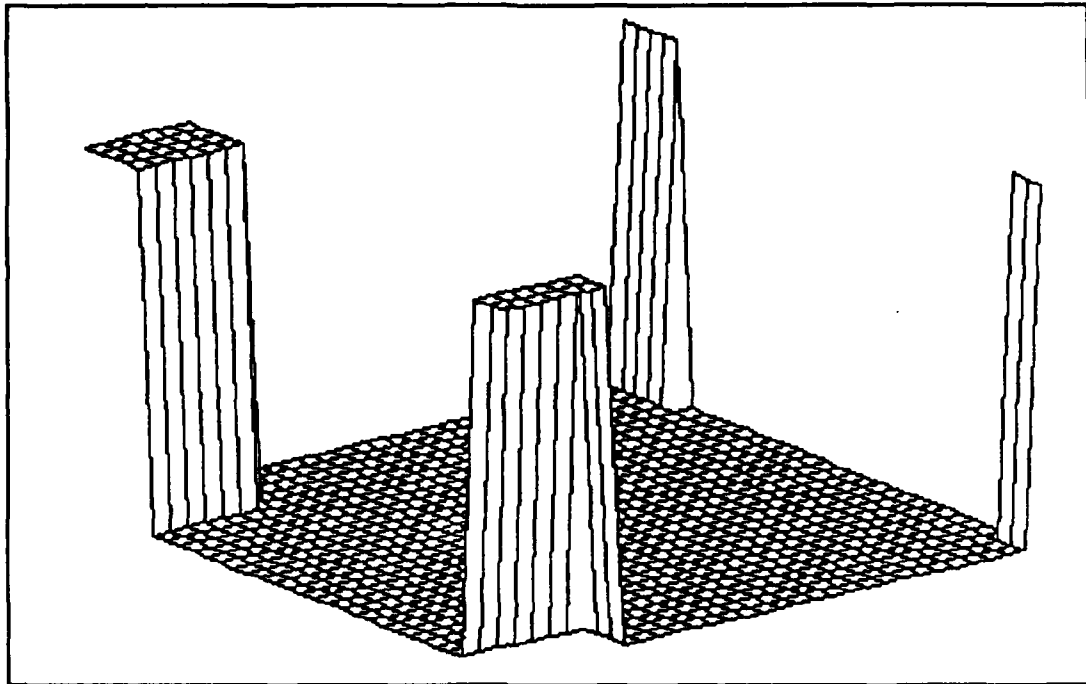


Figure 4.16 Image Frame for Simulation #3 after 95 Iterations.

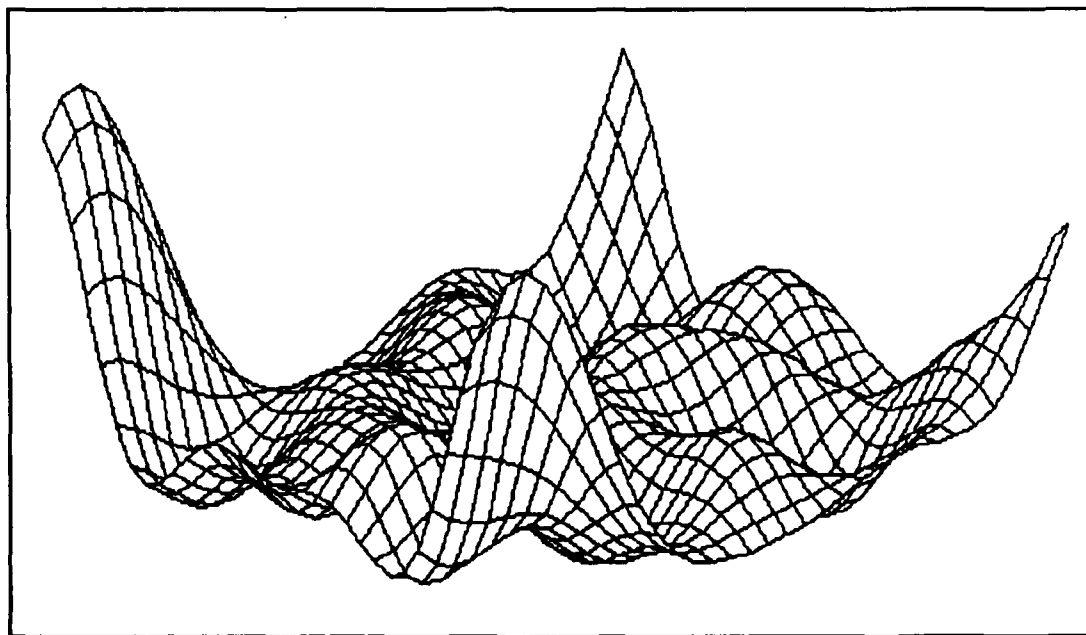


Figure 4.17 Extended Kalman Filter Estimate of Image Frame for Simulation #3 after 95 Iterations.

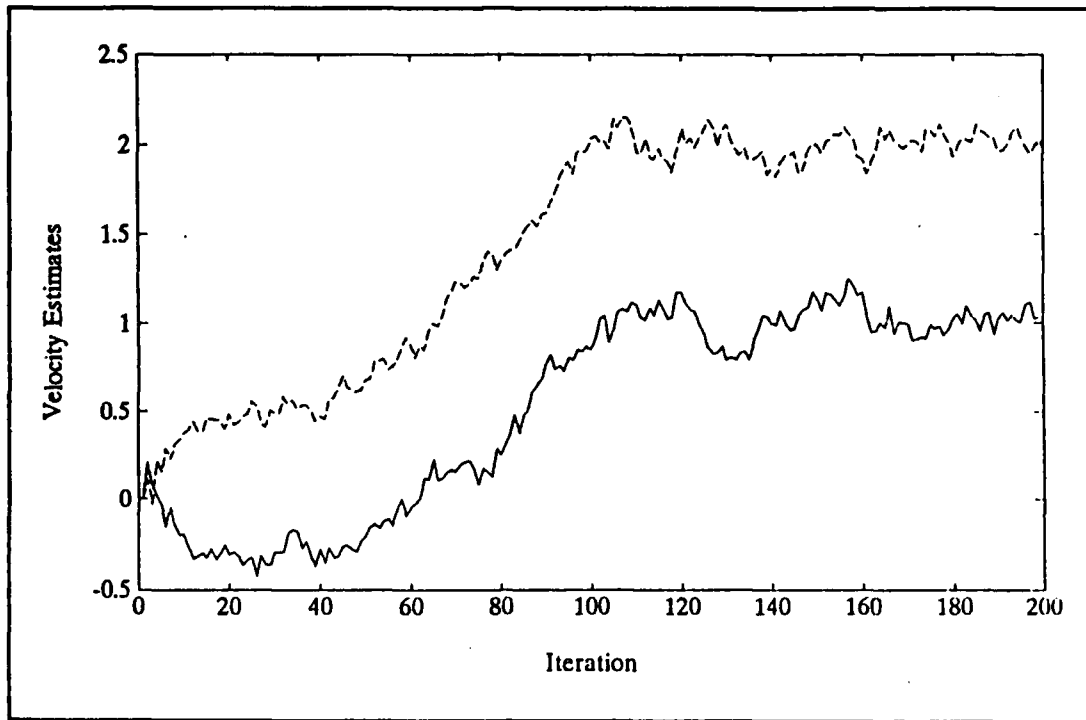


Figure 4.18 Velocity estimates for Simulation #3.

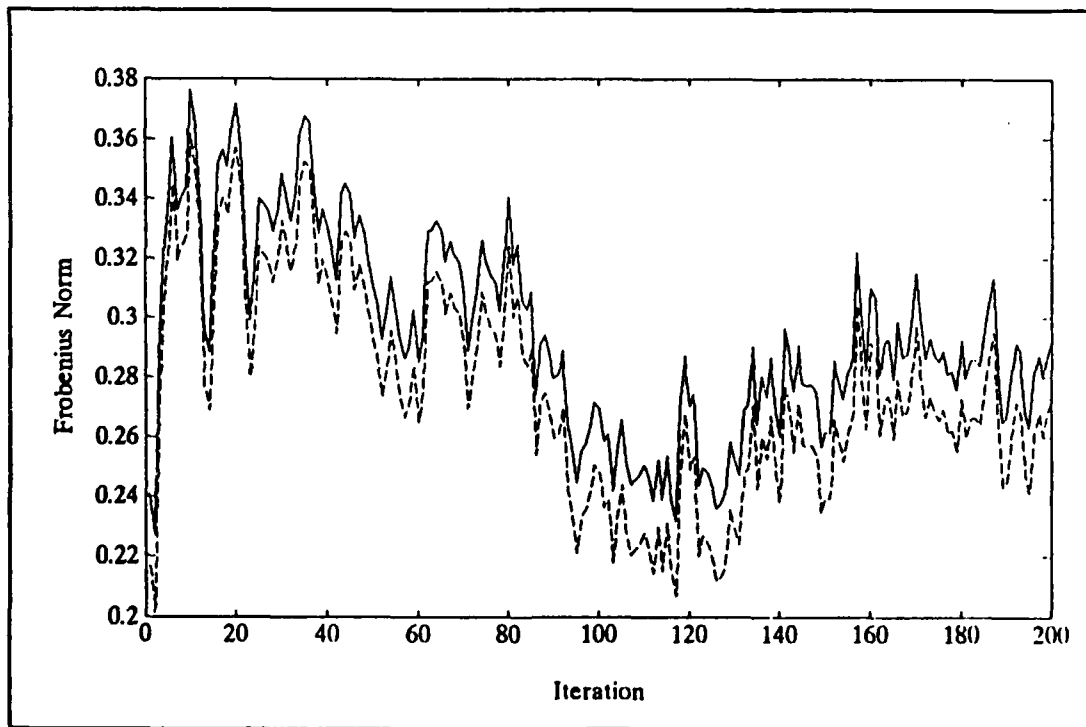


Figure 4.19 Frobenius norm results for Simulation #3.

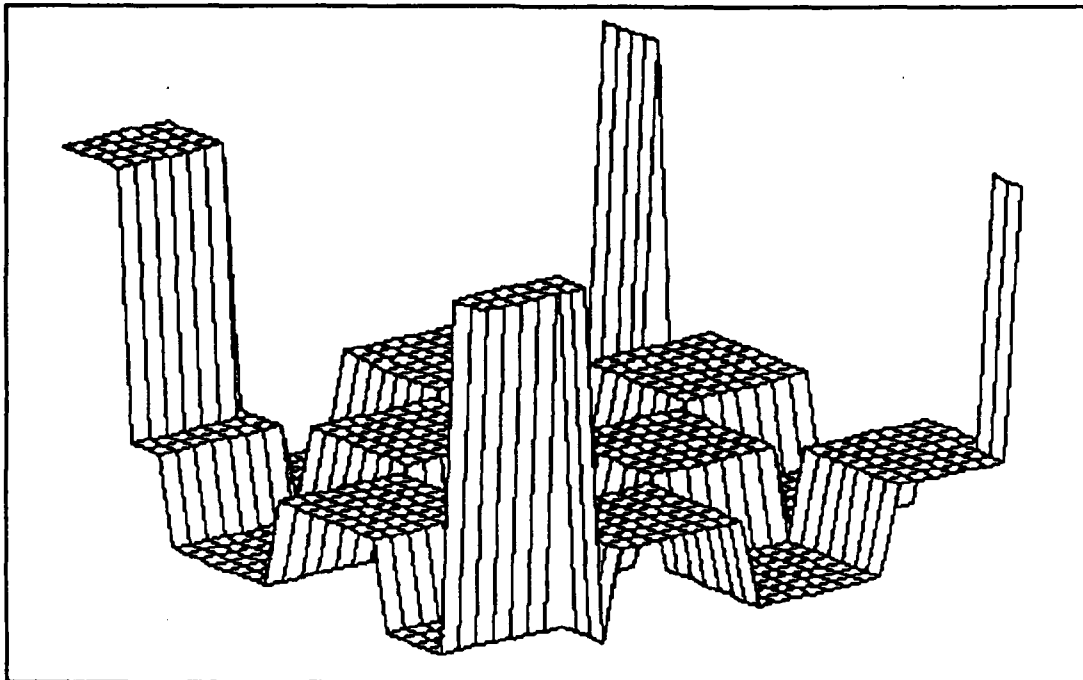


Figure 4.20 Image Frame for Simulation #4 after 95 Iterations.

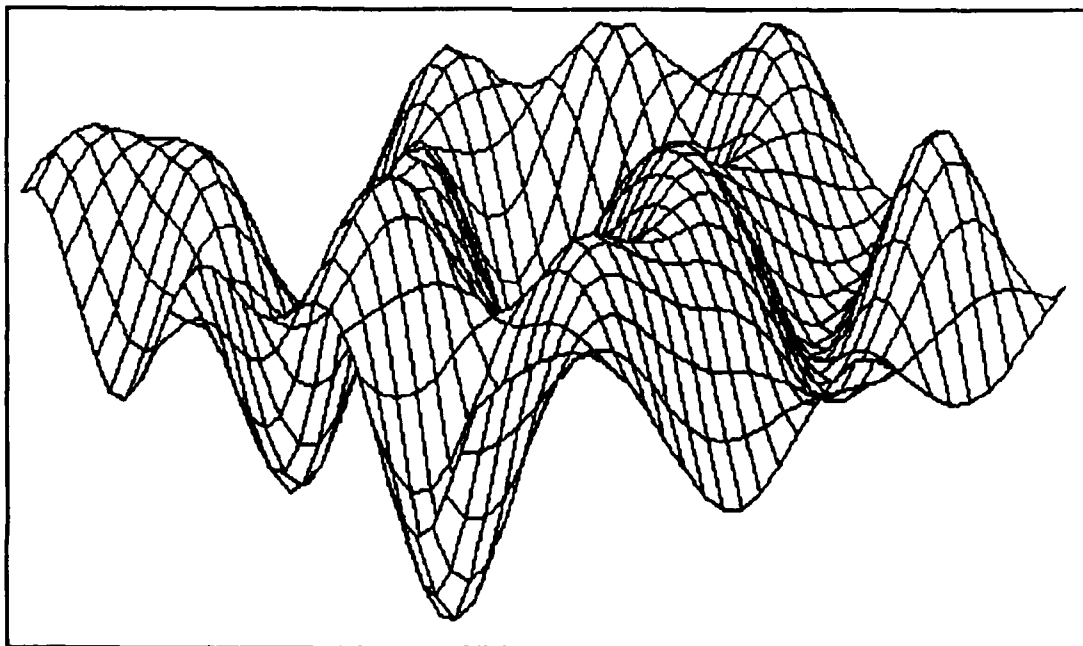


Figure 4.21 Extended Kalman Filter Estimate of Image Frame for Simulation #4 after 95 Iterations.

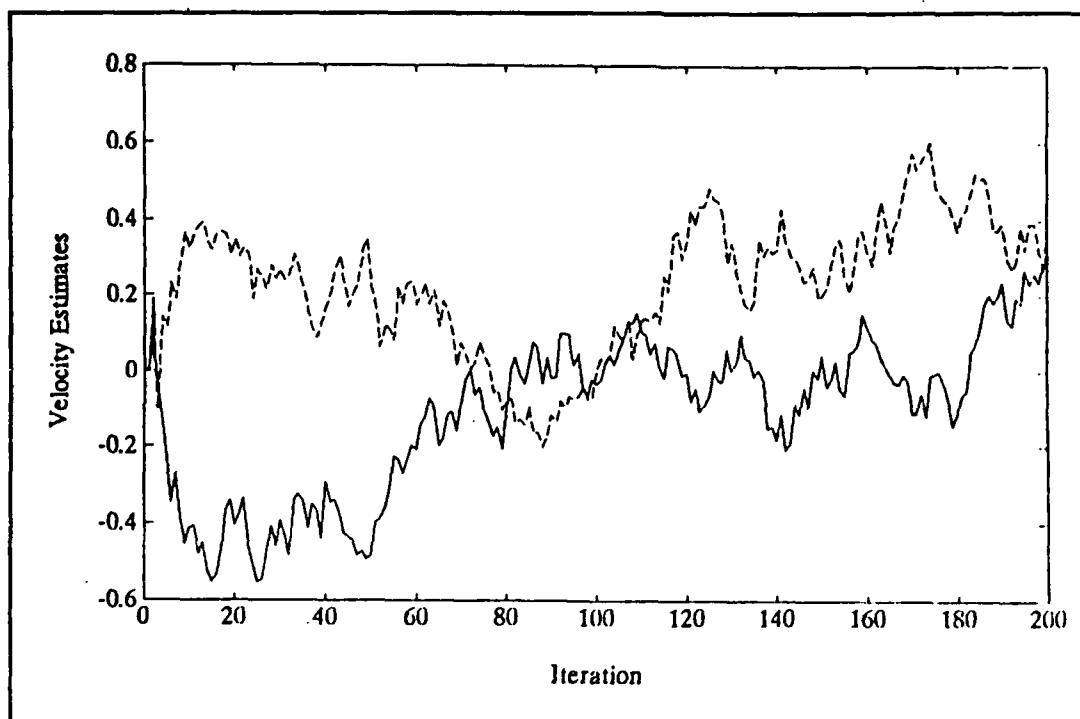


Figure 4.22 Velocity estimates for Simulation #4.

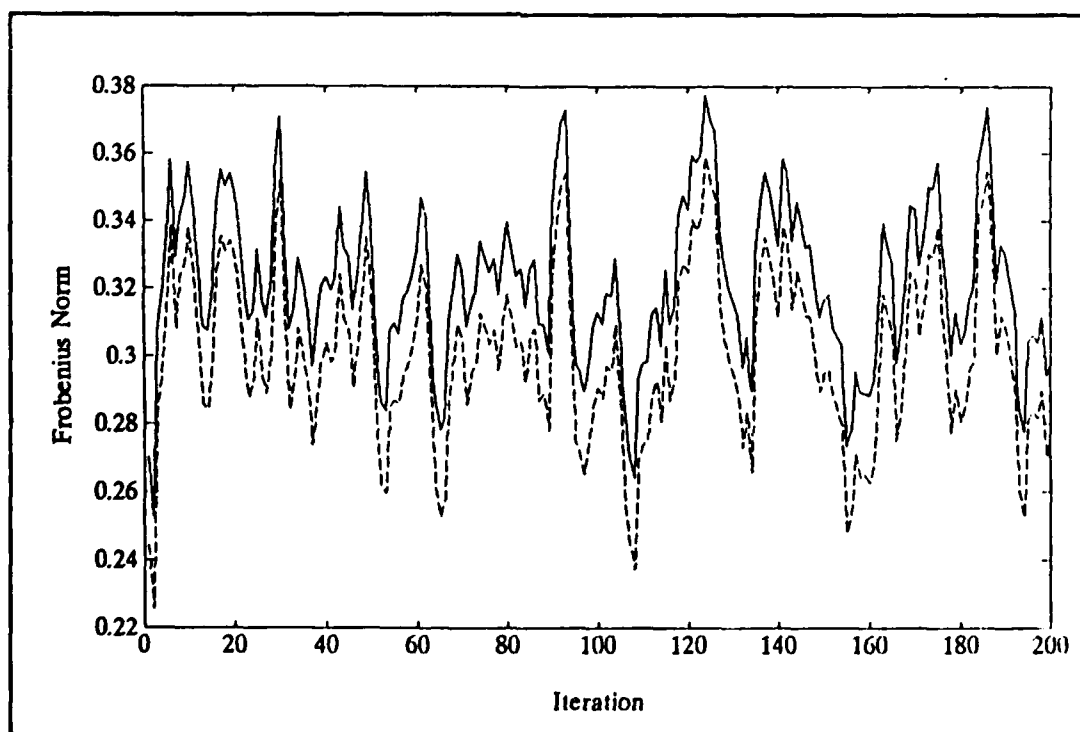


Figure 4.23 Frobenius norm results for Simulation #4.

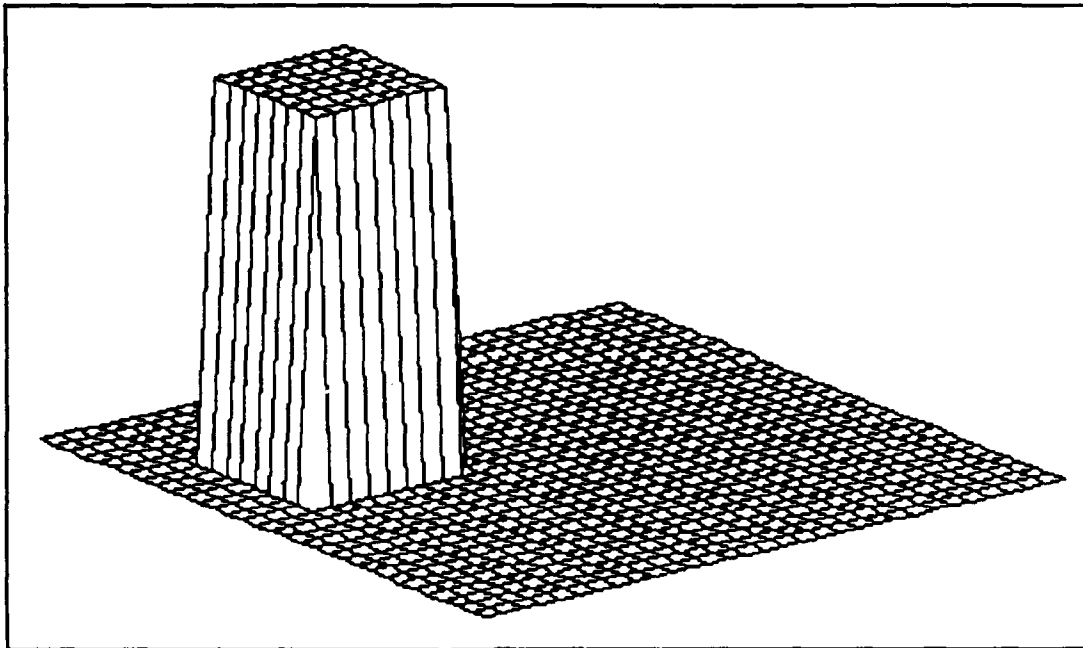


Figure 4.24 Image Frame for Simulation #5 after 100 Iterations.

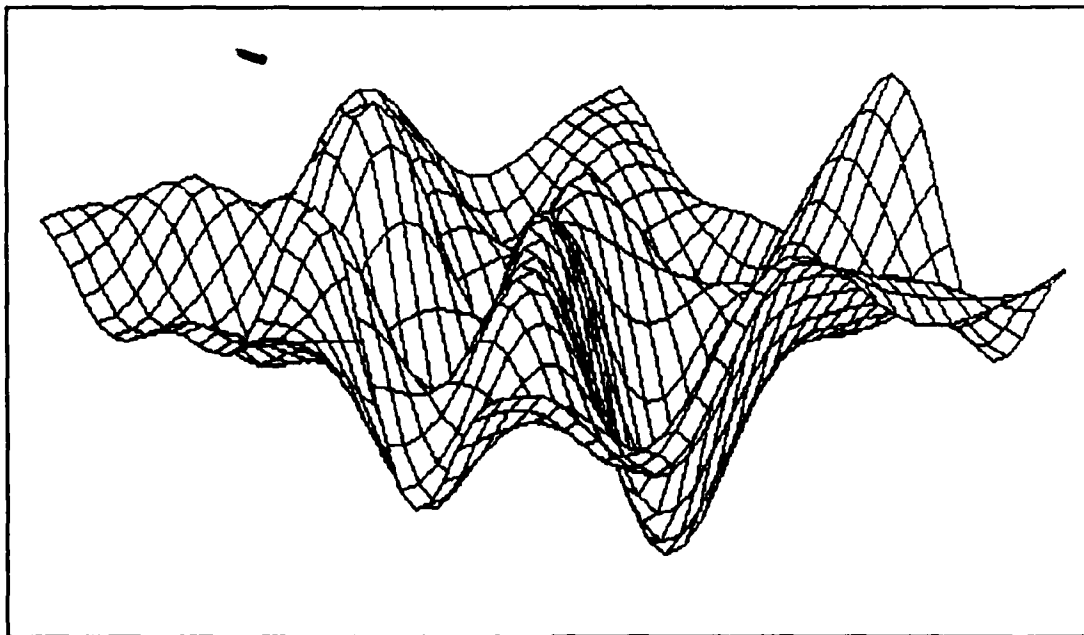


Figure 4.25 Extended Kalman Filter Estimate of Image Frame for Simulation #5 after 100 Iterations.

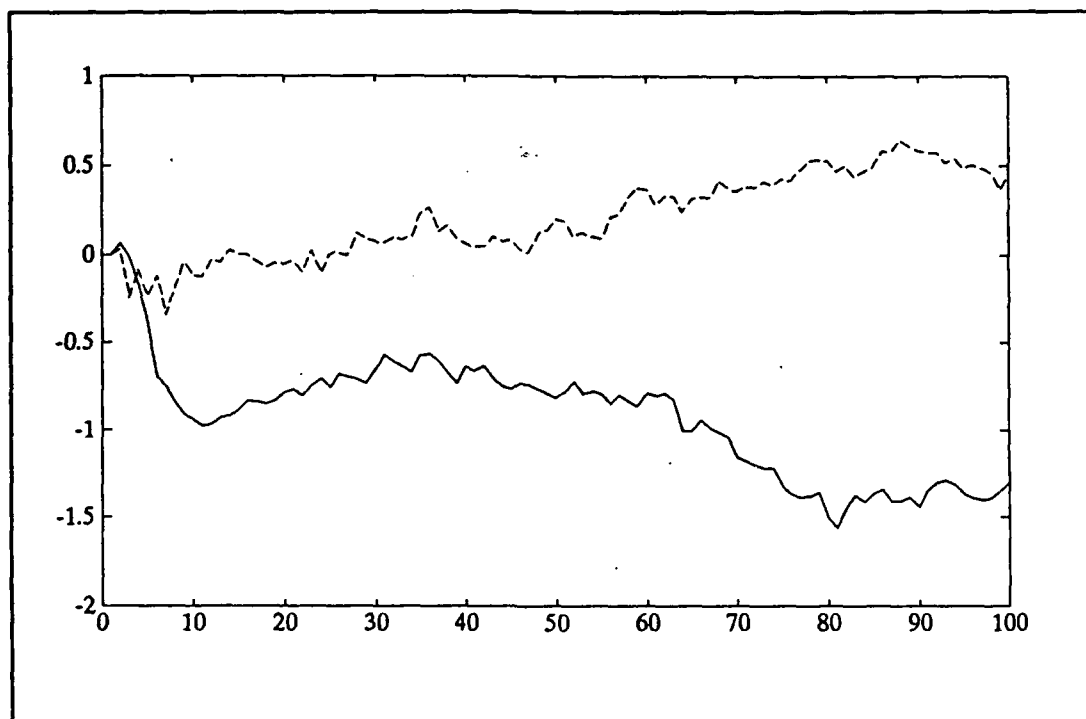


Figure 4.26 Velocity Estimates for Simulation #5.

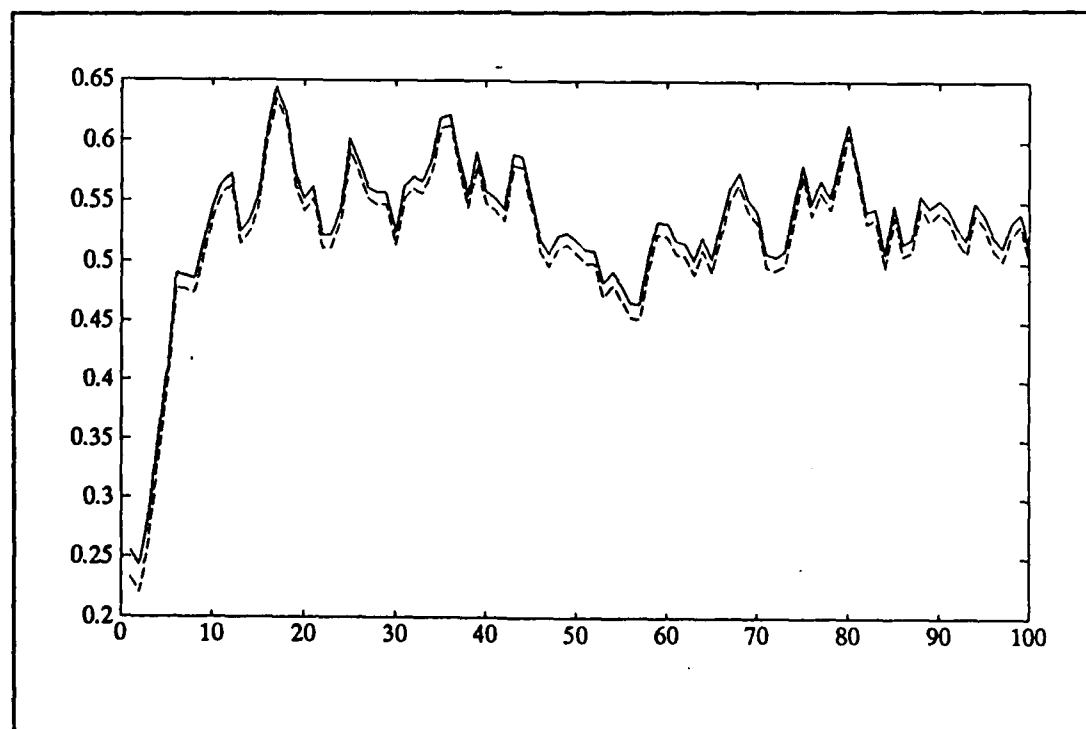


Figure 4.27 Frobenius Norm Results for Simulation #5.

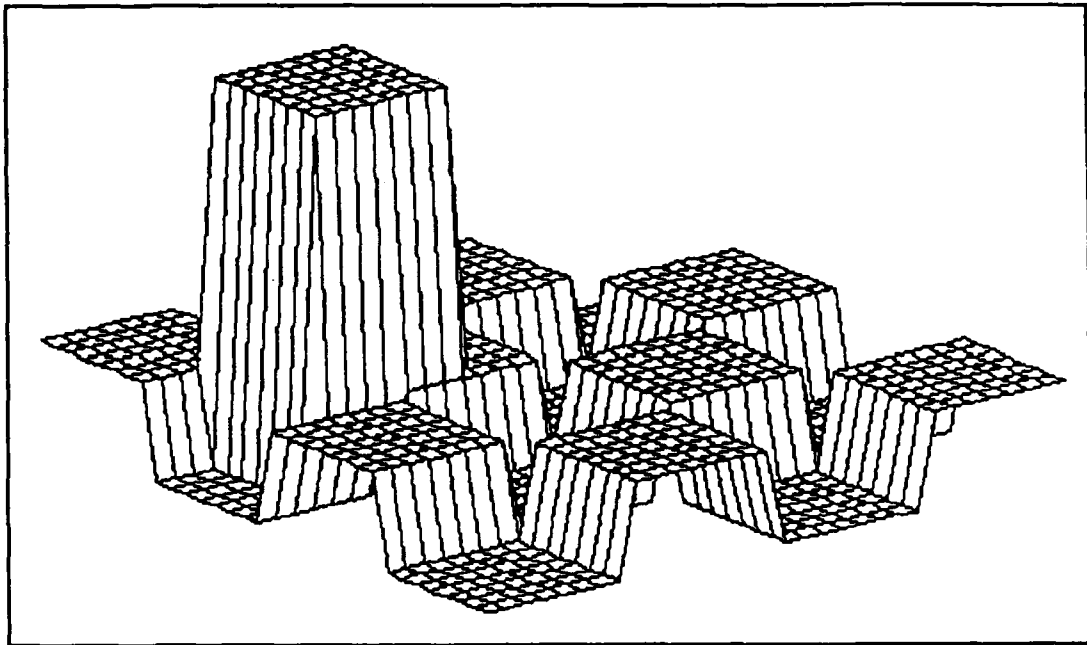


Figure 4.28 Image Frame for Simulation #6 after 100 Iterations.

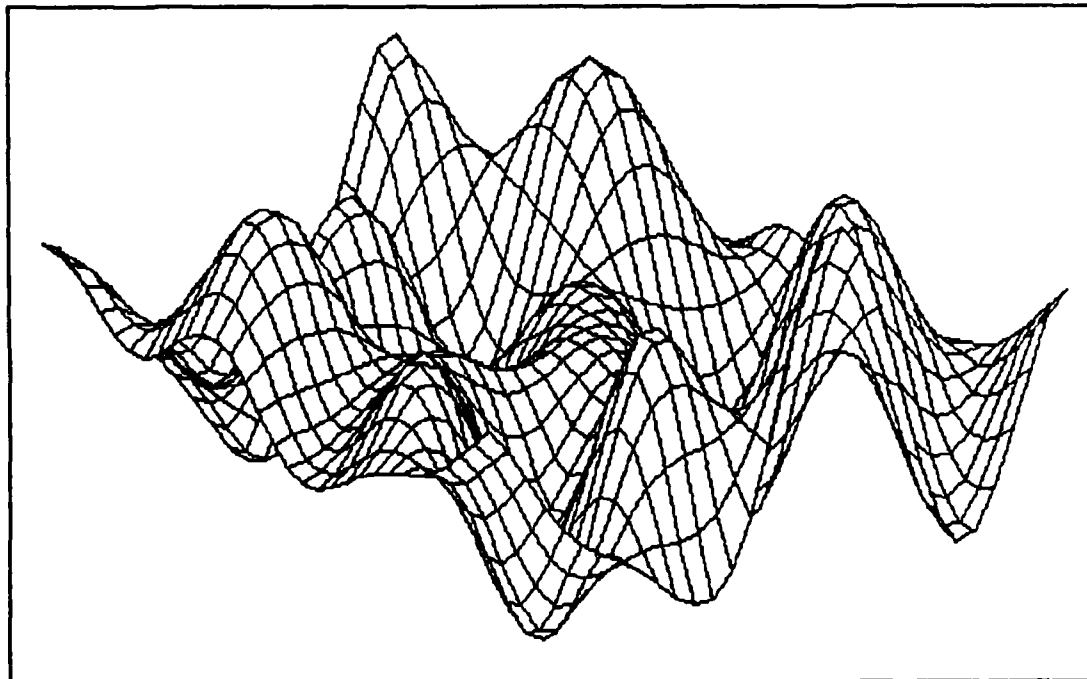


Figure 4.29 Extended Kalman Filter Estimate of Image Frame for Simulation #6 after 100 Iterations.

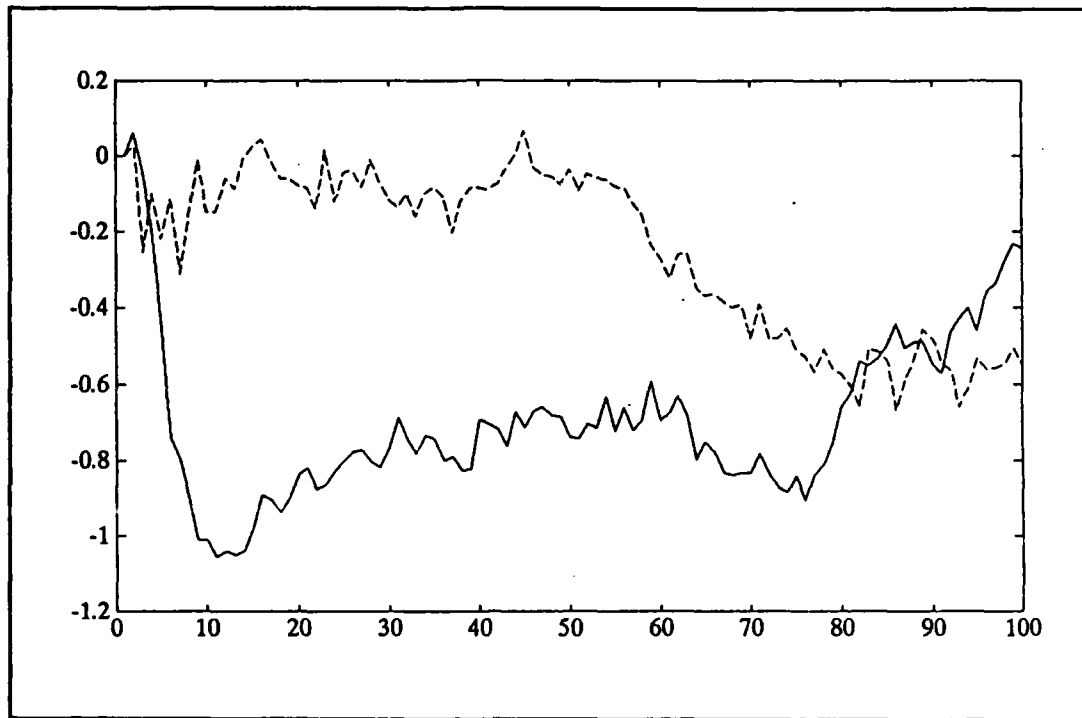


Figure 4.30 Velocity Estimates for Simulation #6.

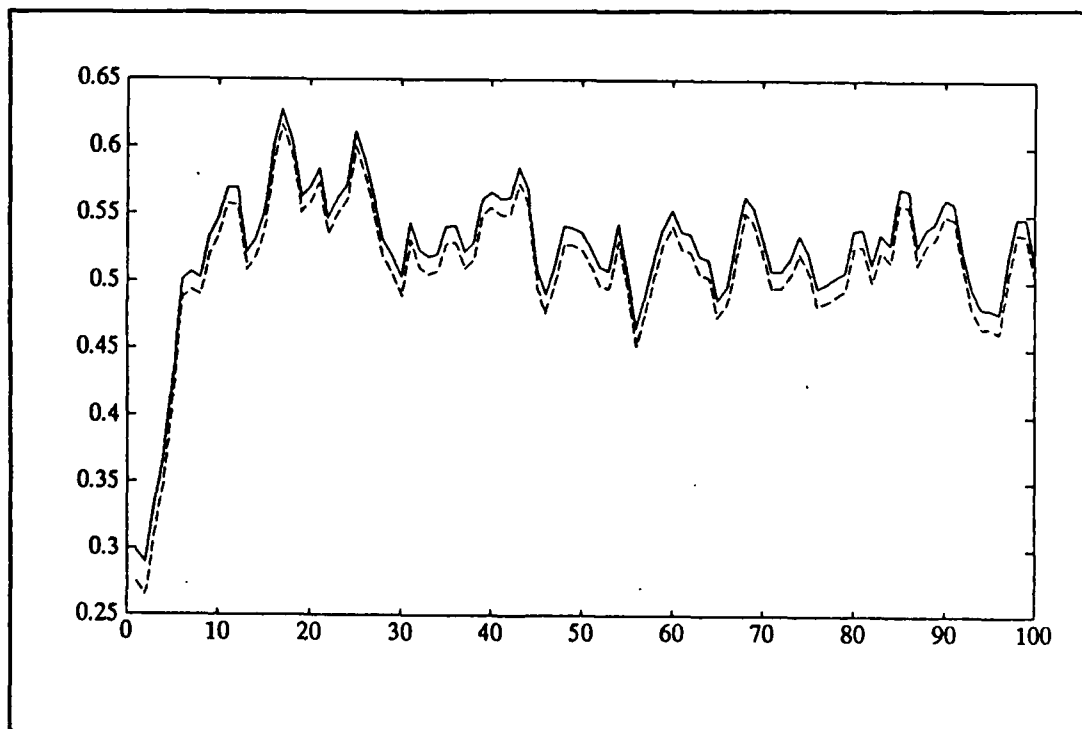


Figure 4.31 Frobenius Norm Results for Simulation #6.

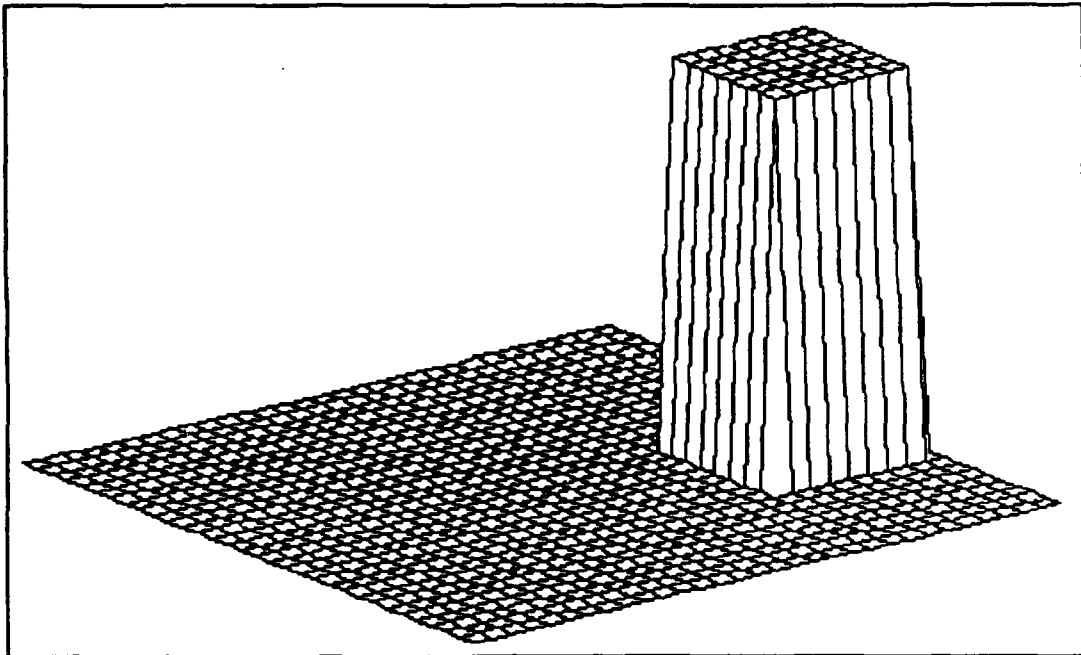


Figure 4.32 Image Frame for Simulation #7 after 45 Iterations.

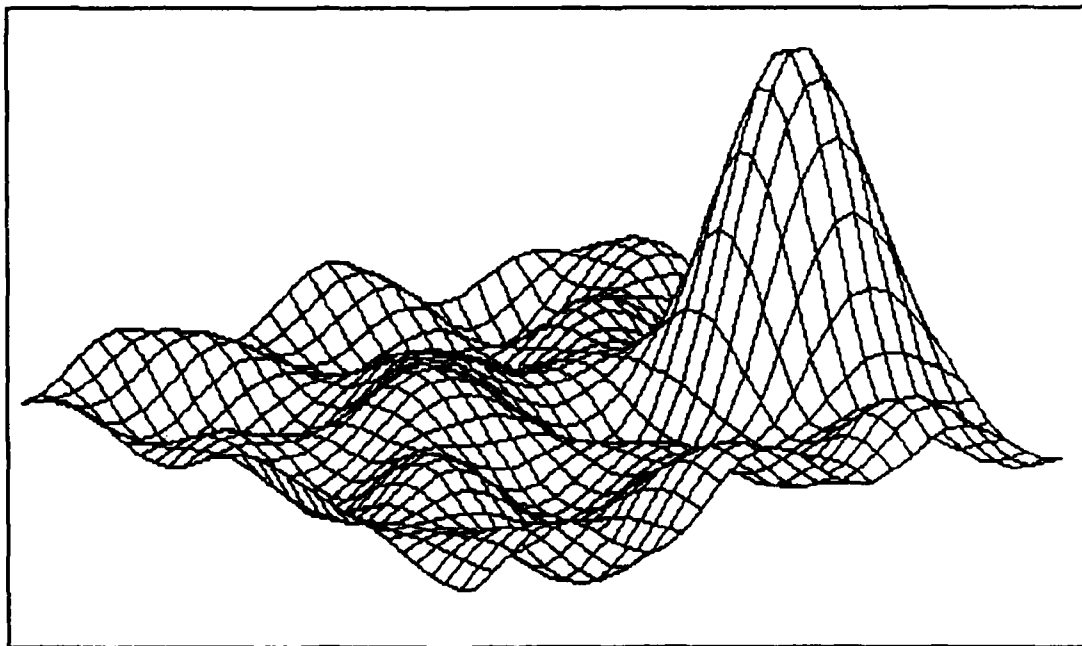


Figure 4.33 Extended Kalman Filter Estimate of Image Frame for Simulation #7 after 45 Iterations.

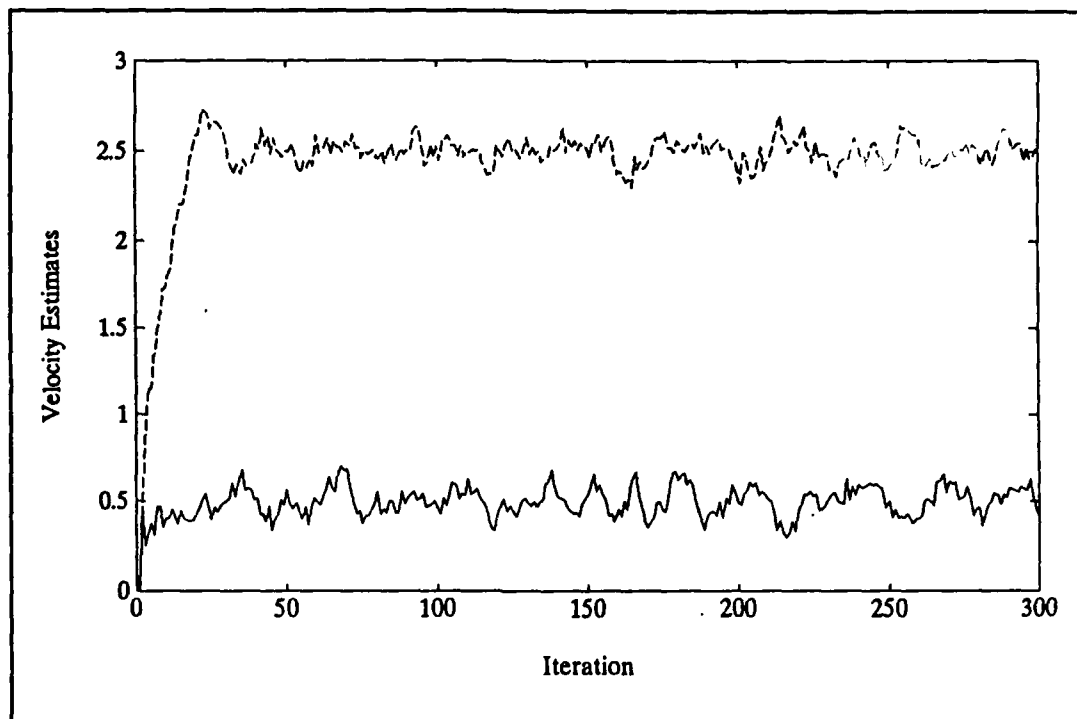


Figure 4.34 Velocity Estimates for Simulation #7.

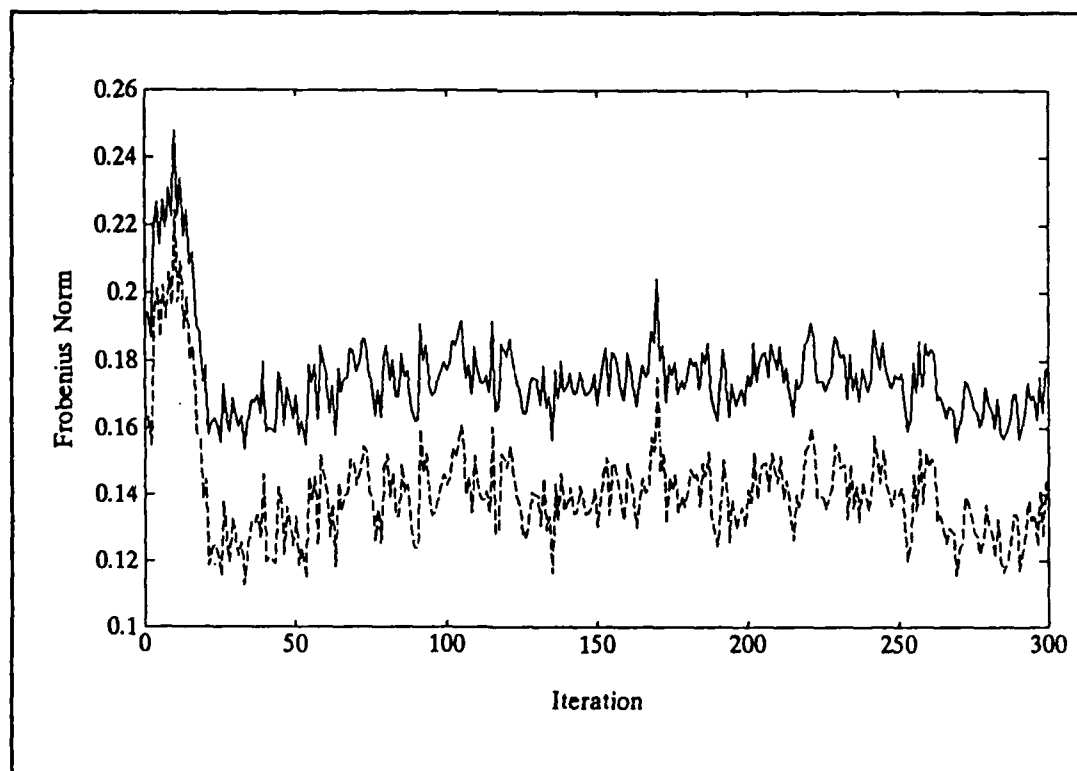


Figure 4.35 Frobenius Norm Results for Simulation #7.

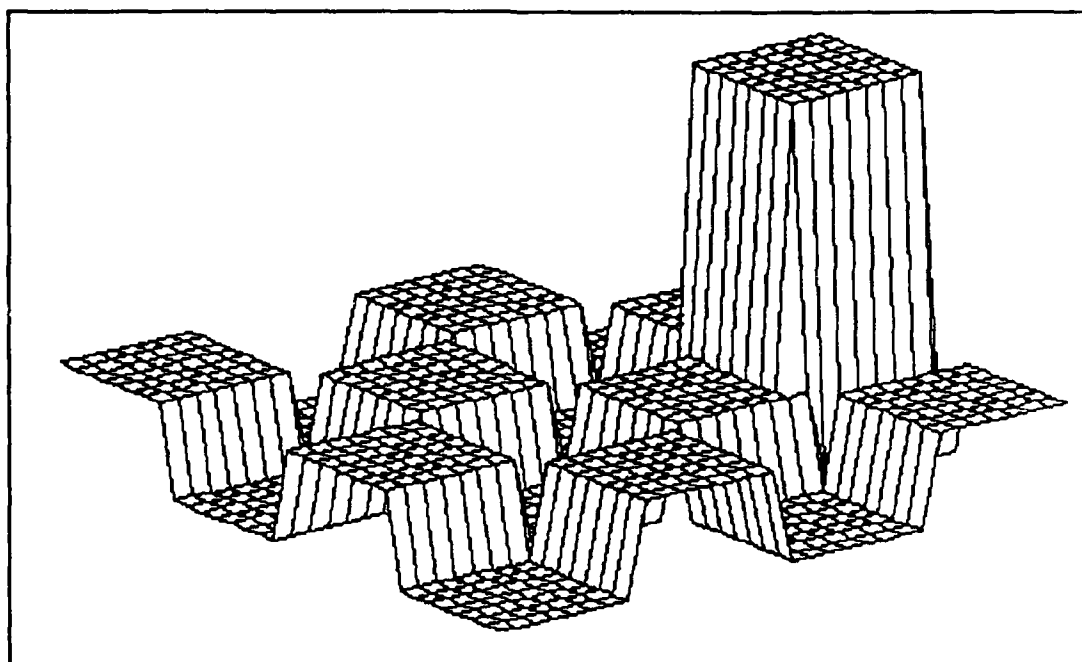


Figure 4.36 Image Frame for Simulation #8 after 95 Iterations.

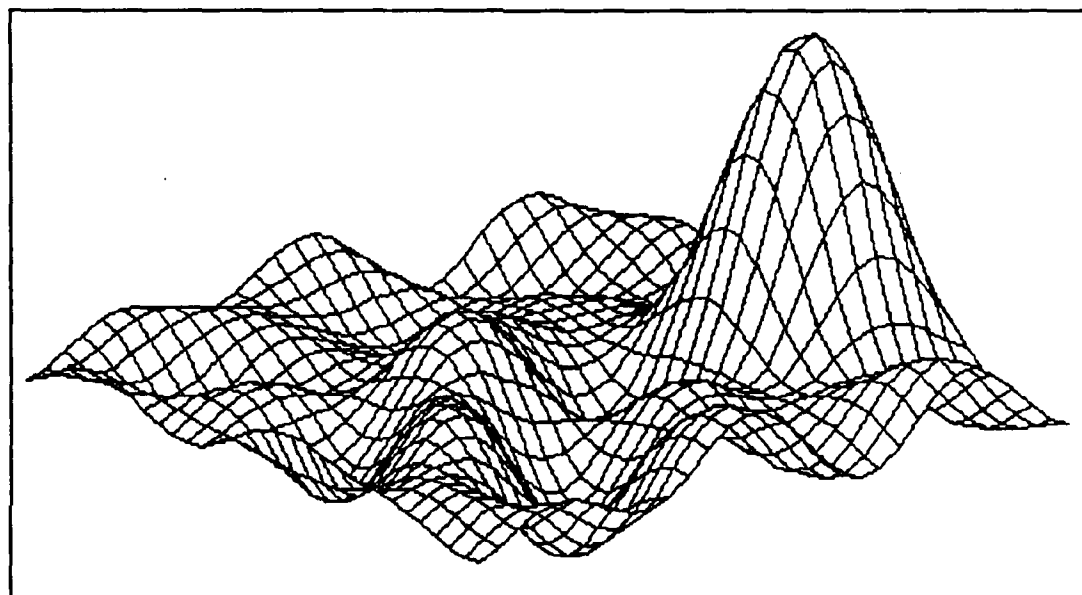


Figure 4.37 Extended Kalman Filter Estimate of Image Frame for Simulation #8 after 45 Iterations.



Figure 4.38 Velocity Estimates for Simulation #8.

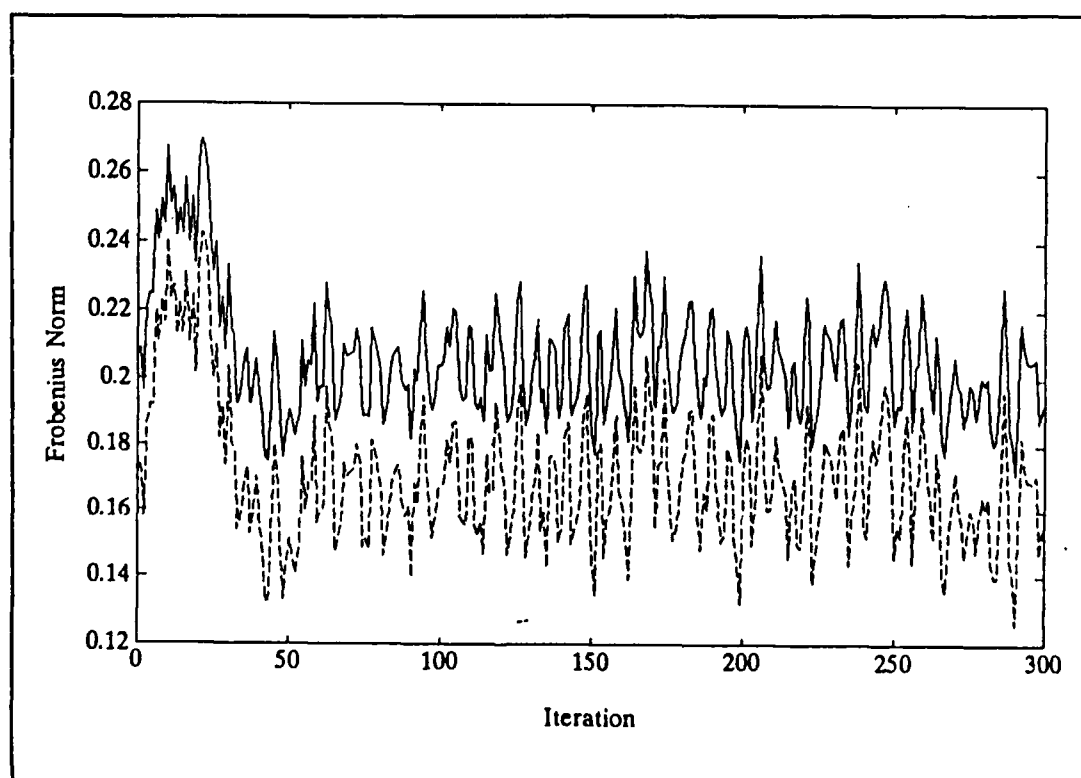


Figure 4.39 Frobenius Norm Results for Simulation #8.

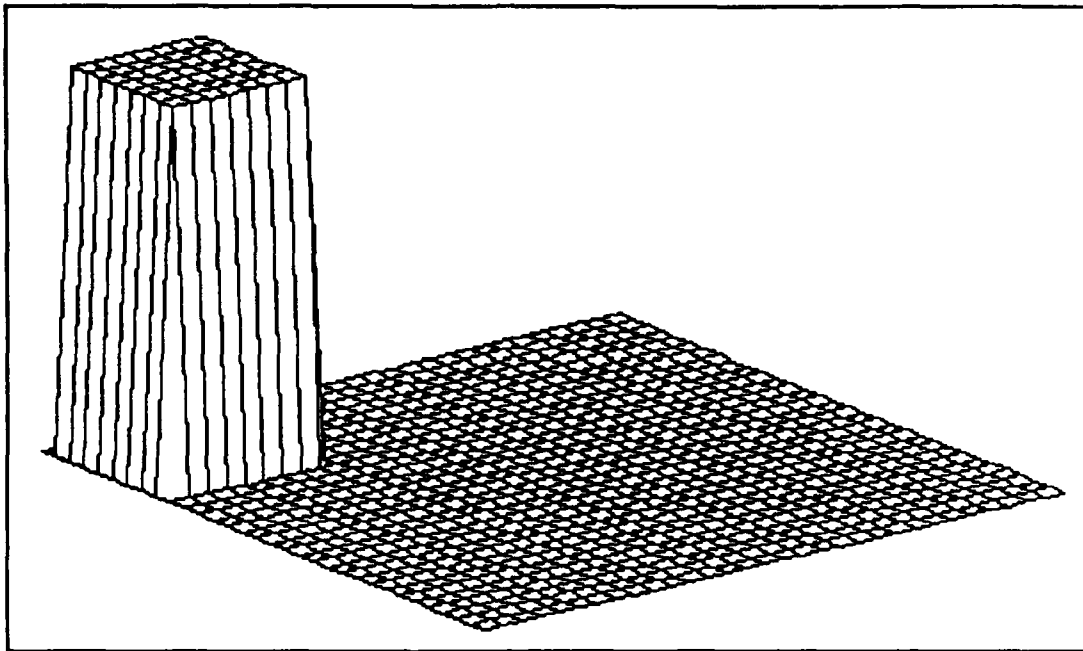


Figure 4.40 Image Frame for Simulation #9 after 65 Iterations.

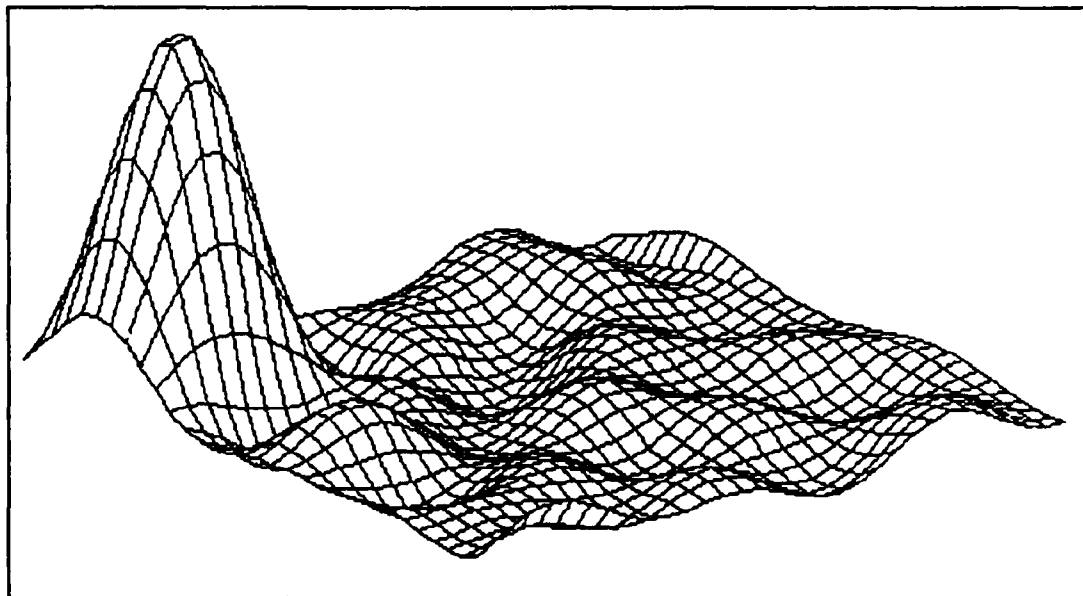


Figure 4.41 Extended Kalman Filter Estimate of Image Frame for Simulation #9 after 65 Iterations.

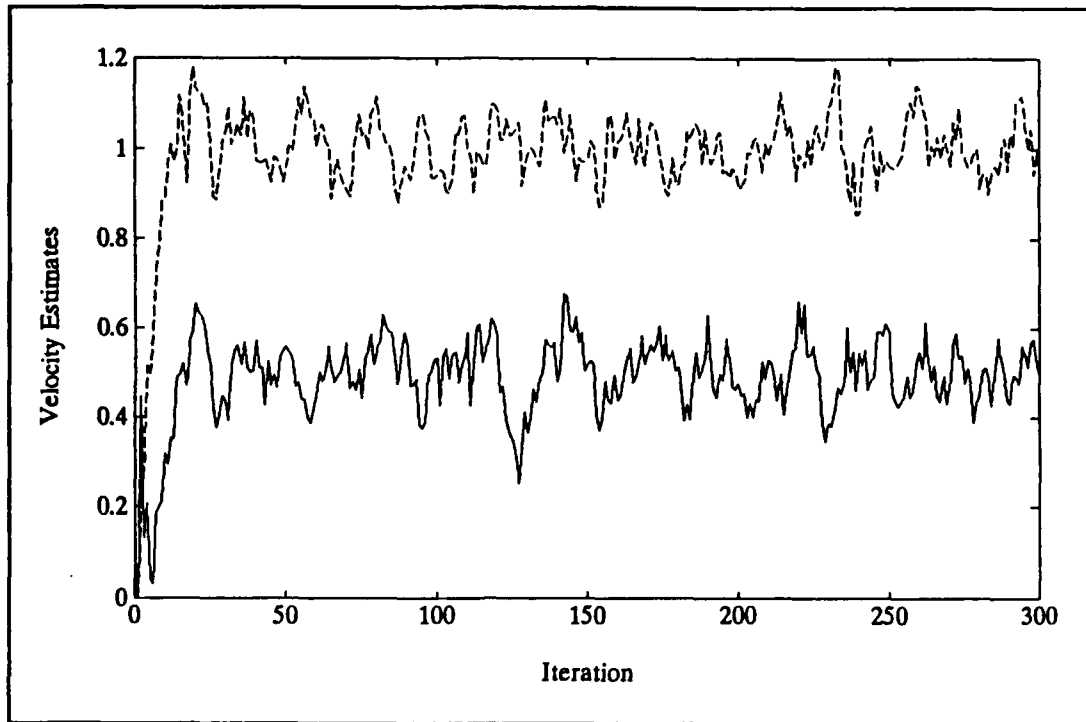


Figure 4.42 Velocity Estimates for Simulation #9.

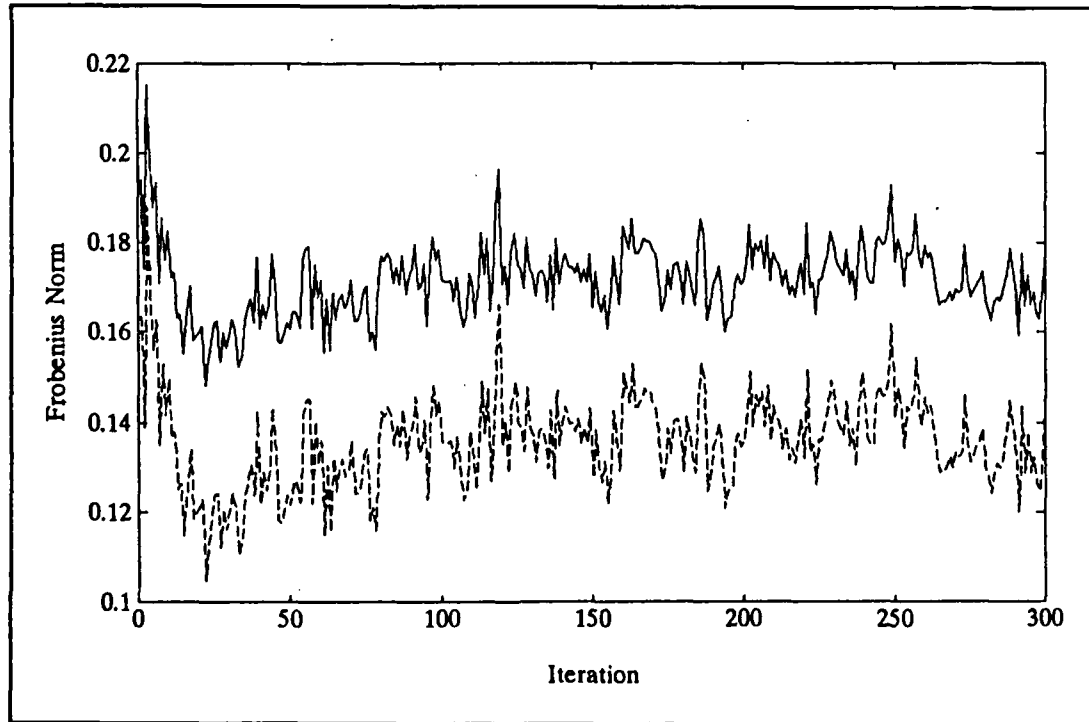


Figure 4.43 Frobenius Norm Results for Simulation #9.

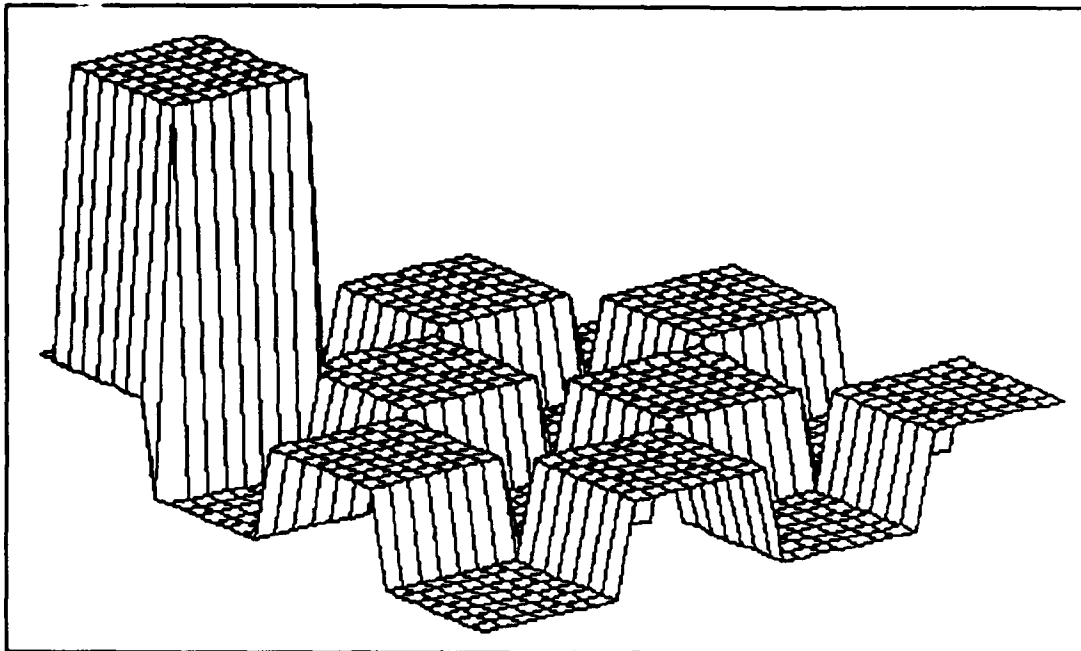


Figure 4.44 Image Frame for Simulation #10 after 65 Iterations.

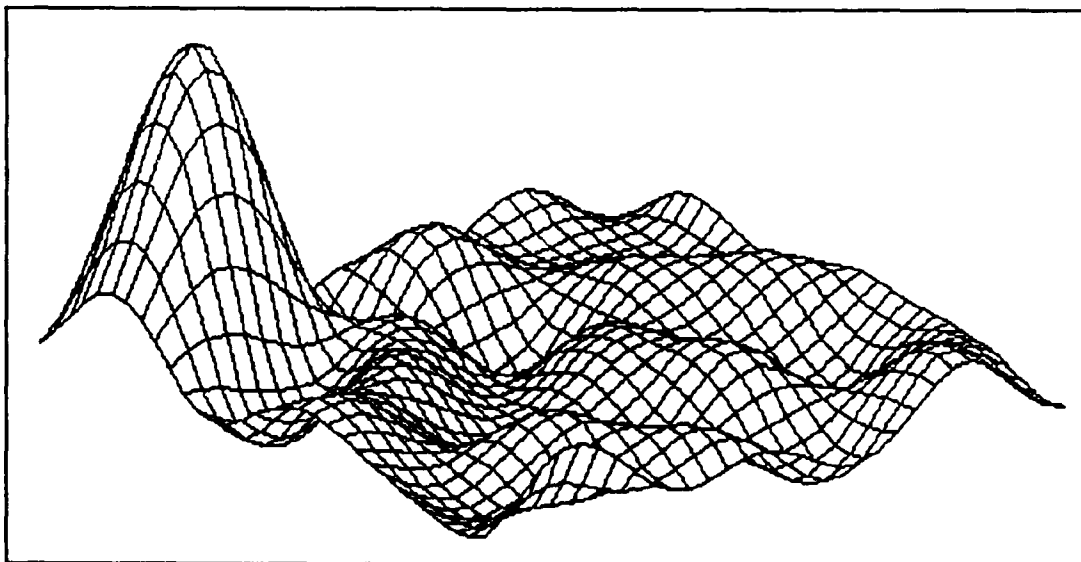


Figure 4.45 Extended Kalman Filter Estimate of Image Frame for Simulation #10 after 65 Iterations.

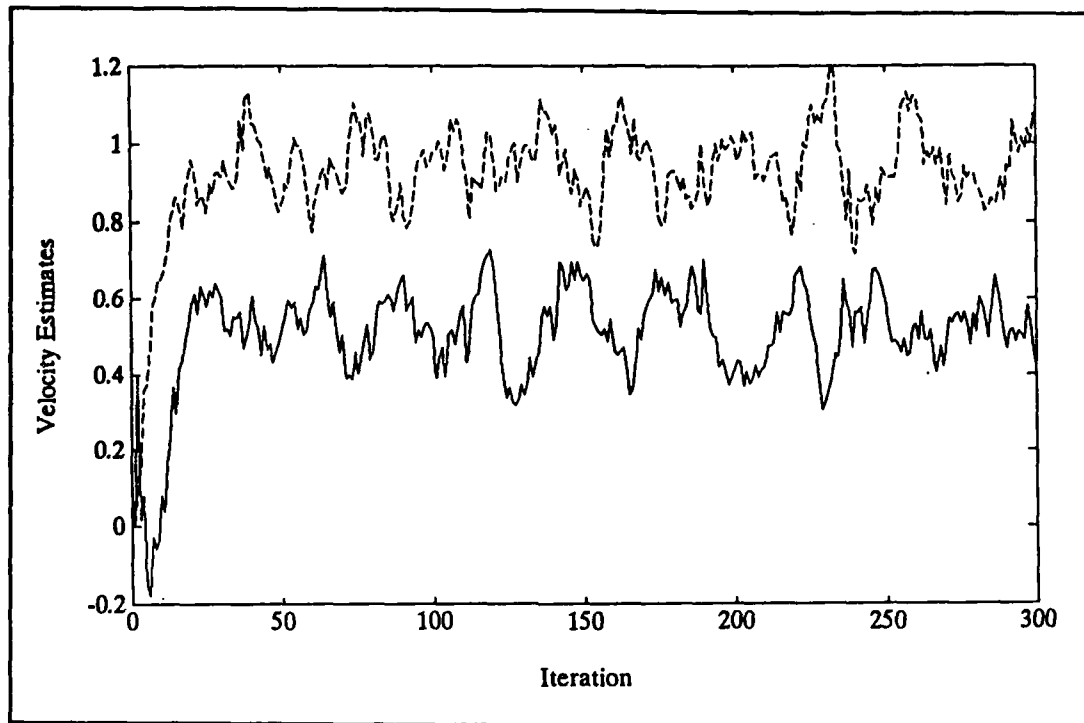


Figure 4.46 Velocity Estimates for Simulation #10.

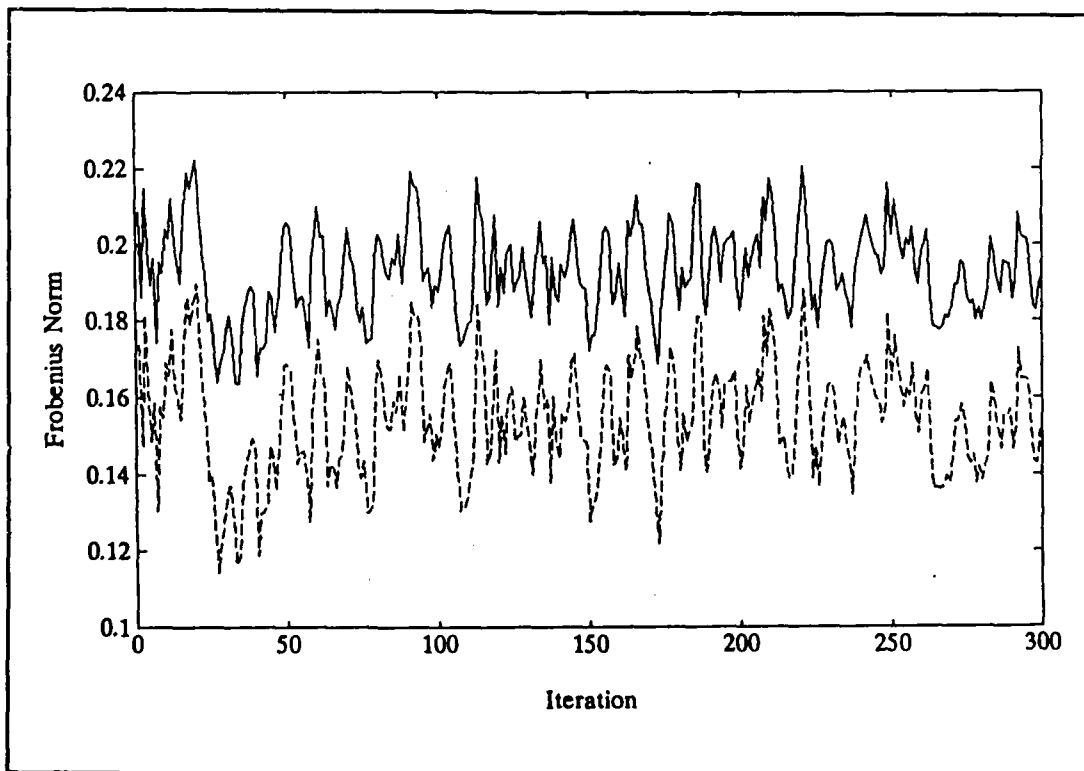


Figure 4.47 Frobenius Norm Results for Simulation #10.

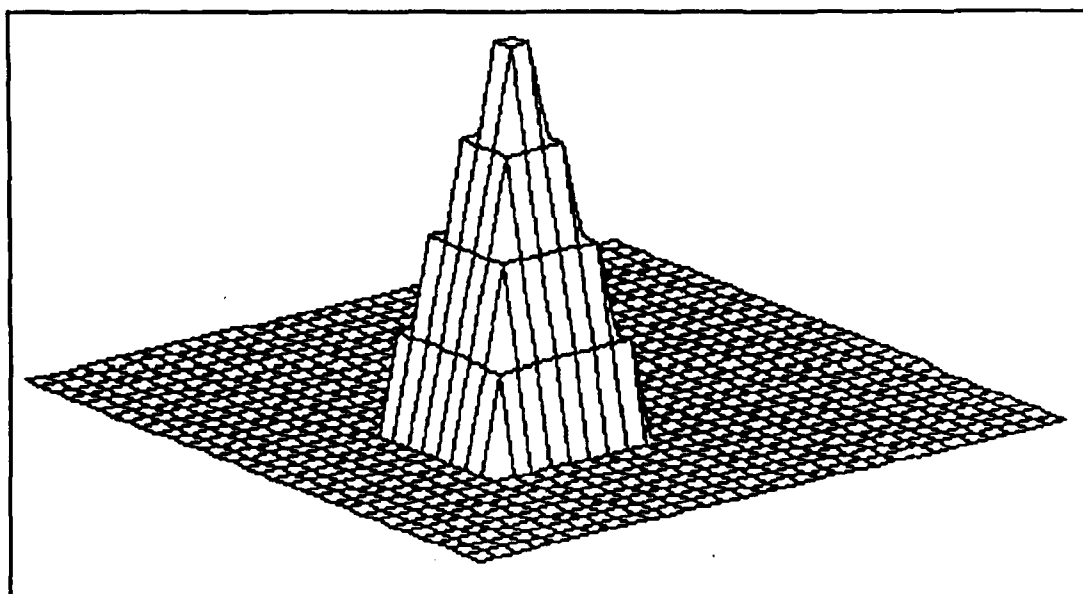


Figure 4.48 Image Frame for Simulation #1 after 40 Iterations.

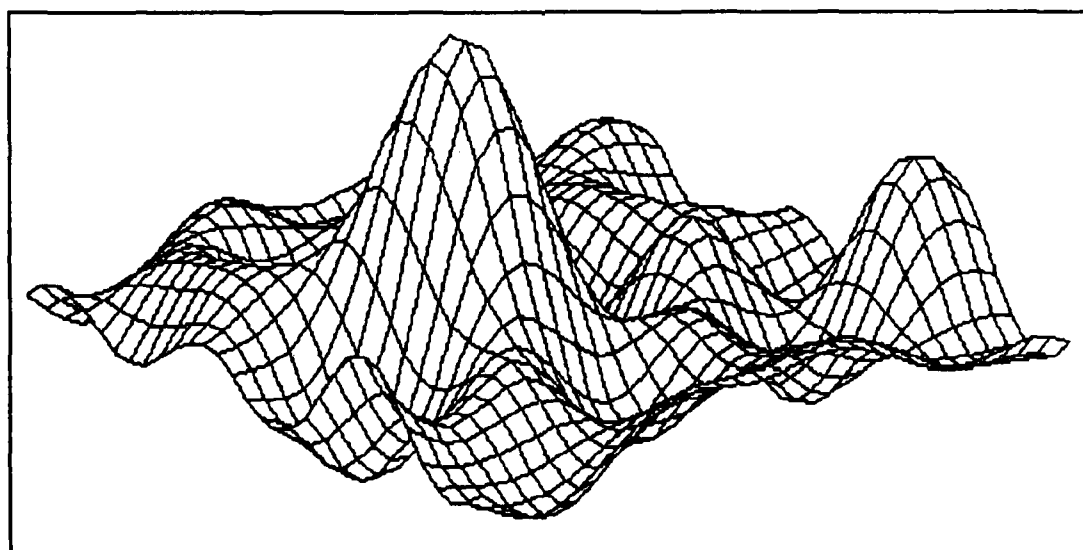


Figure 4.49 Extended Kalman Filter Estimate of Image Frame for Simulation #1 after 40 Iterations.

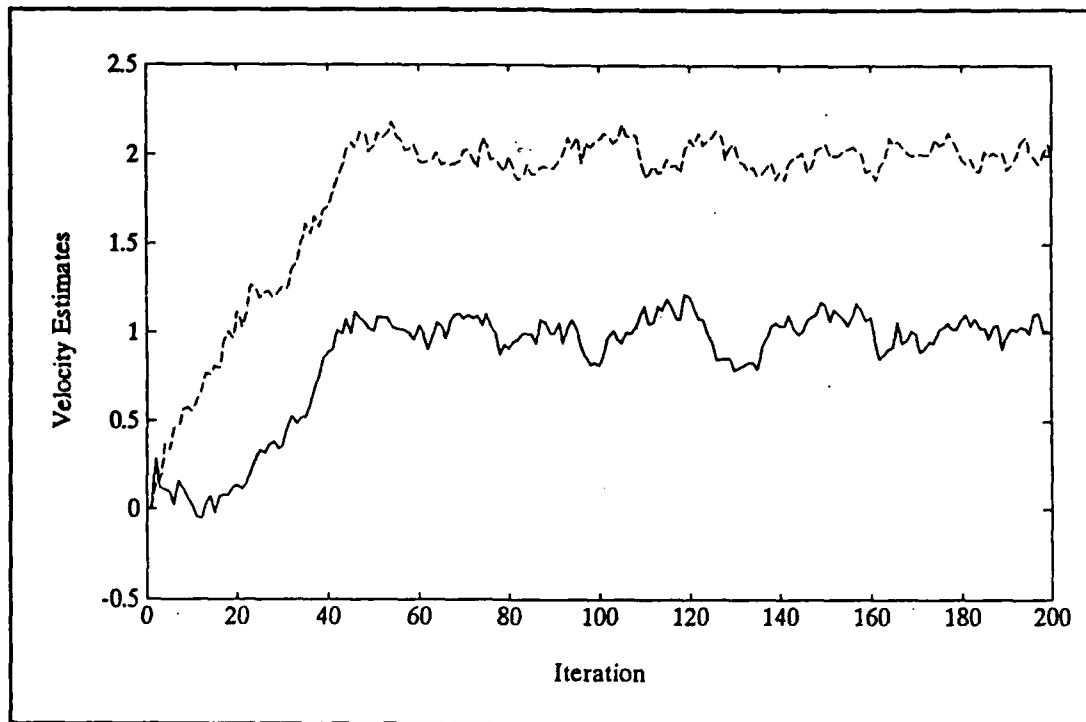


Figure 4.50 Velocity Estimates for Simulation #1.

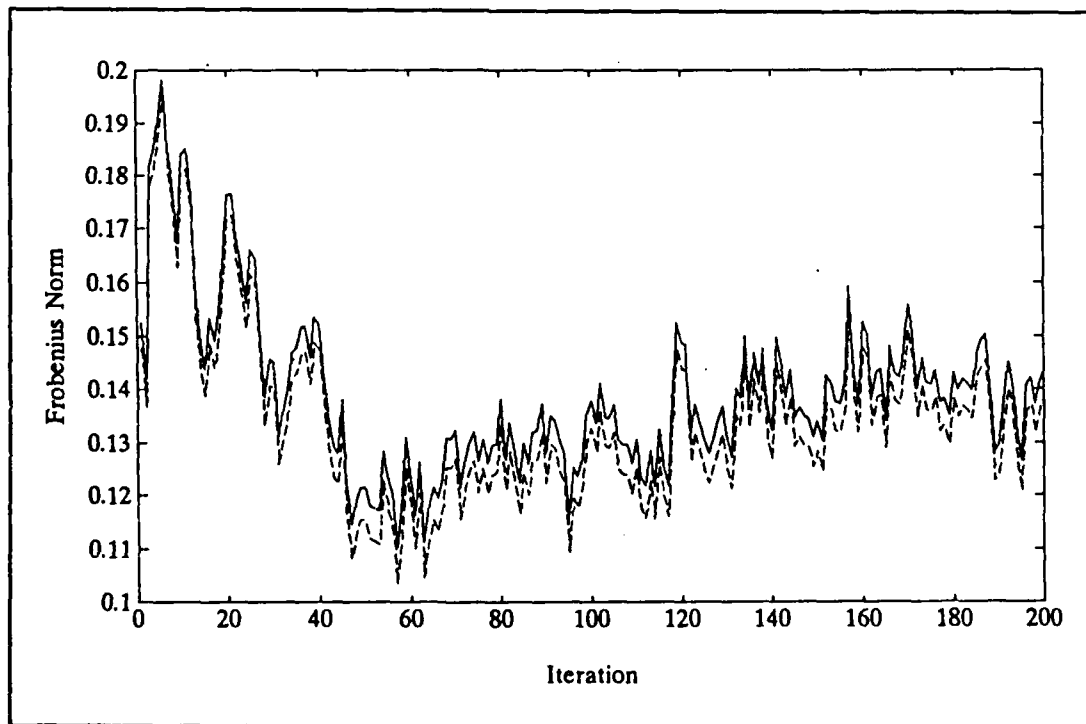


Figure 4.51 Frobenius Norm Results for Simulation #1.

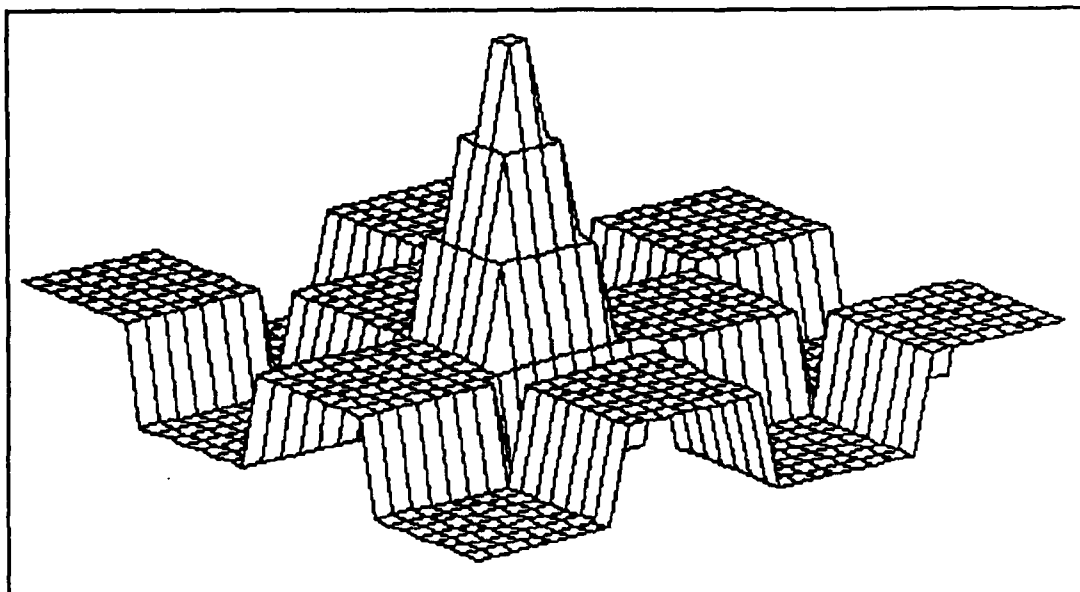


Figure 4.52 Image Frame for Simulation #2 after 40 Iterations.

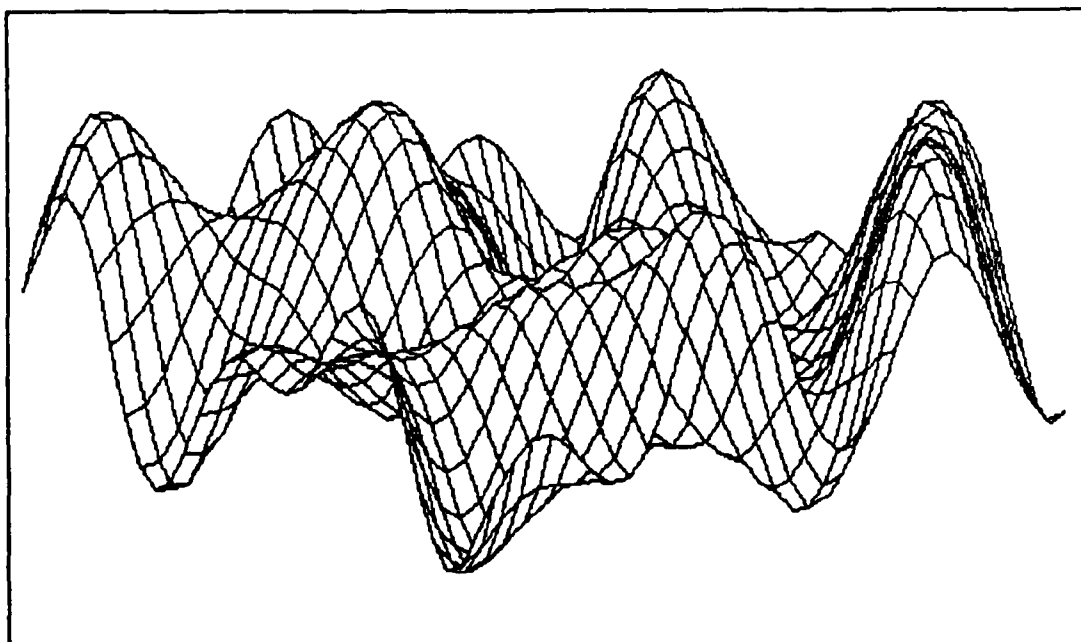


Figure 4.53 Extended Kalman Filter Estimate of Image Frame for Simulation #2 after 40 Iterations.

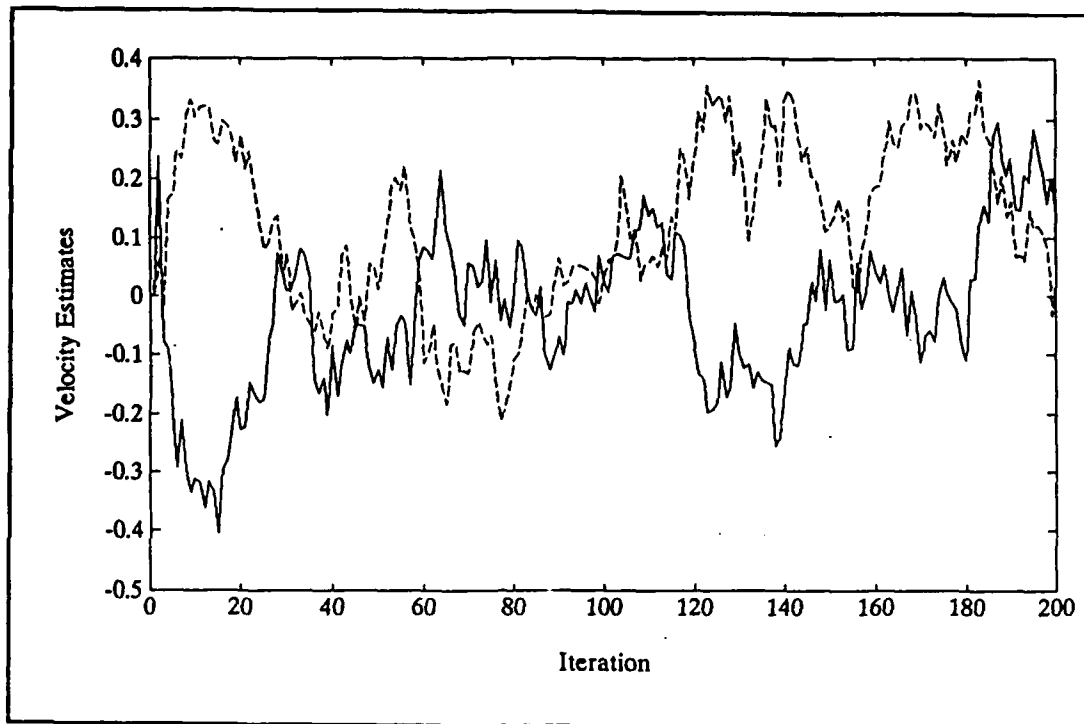


Figure 4.54 Velocity Estimates for Simulation #2.

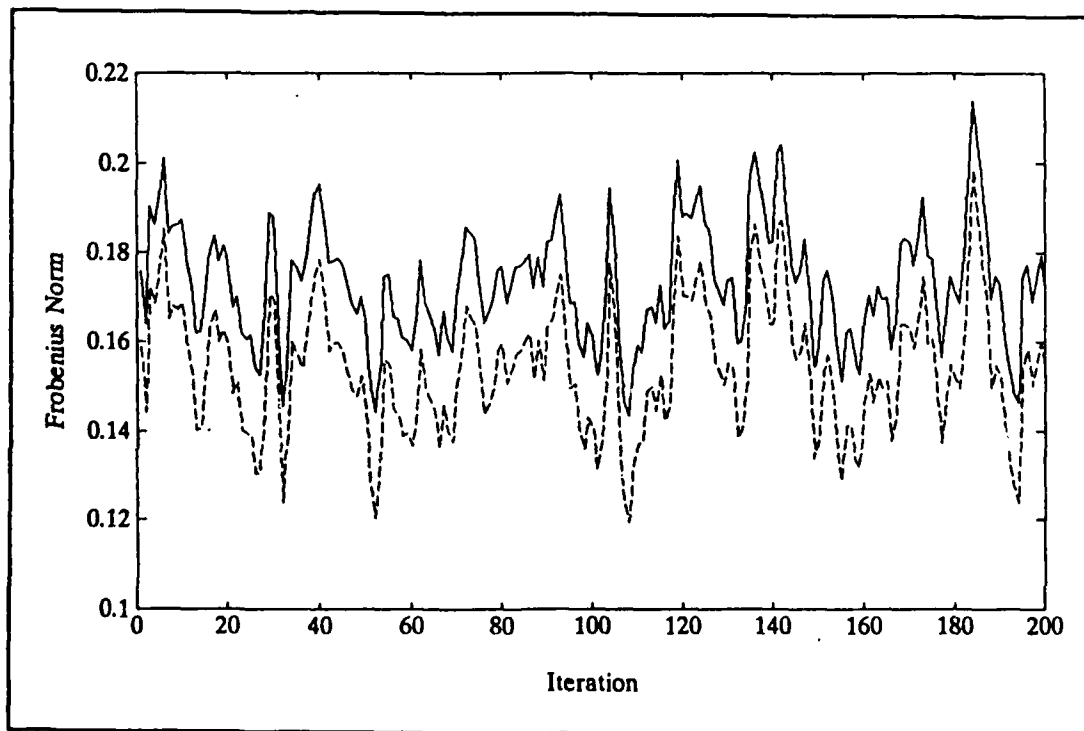


Figure 4.55 Frobenius Norm Results for Simulation #2.

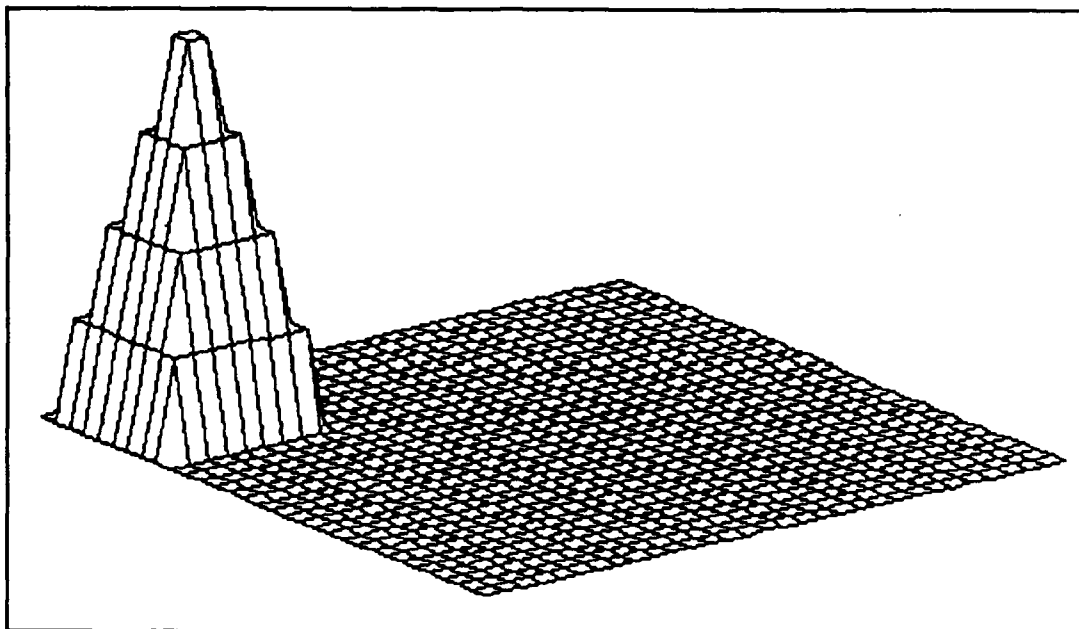


Figure 4.56 Image Frame for Simulation #3 after 65 Iterations.

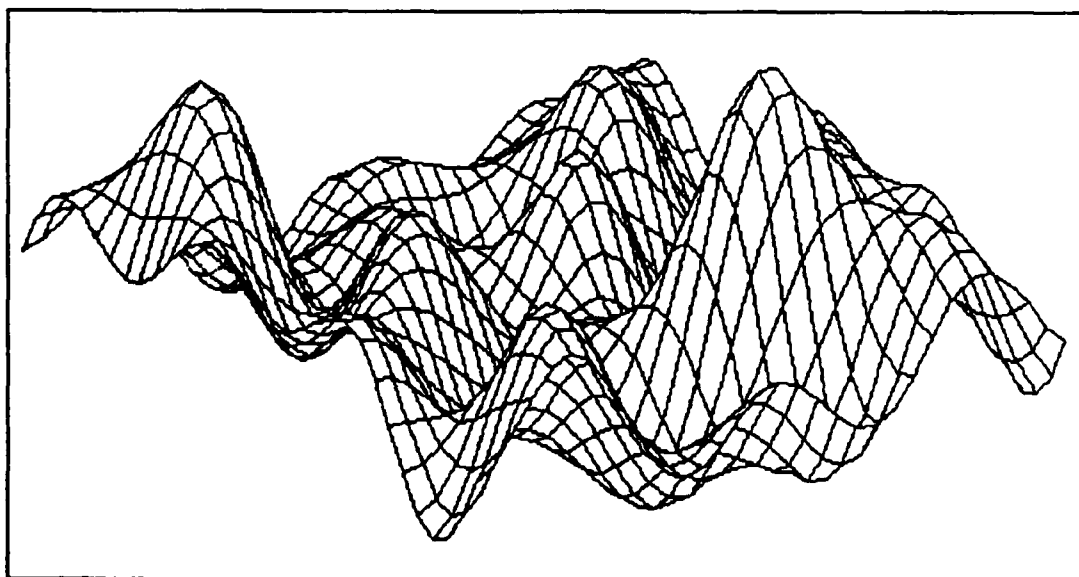


Figure 4.57 Extended Kalman Filter Estimate of Image Frame for Simulation #3 after 65 Iterations.

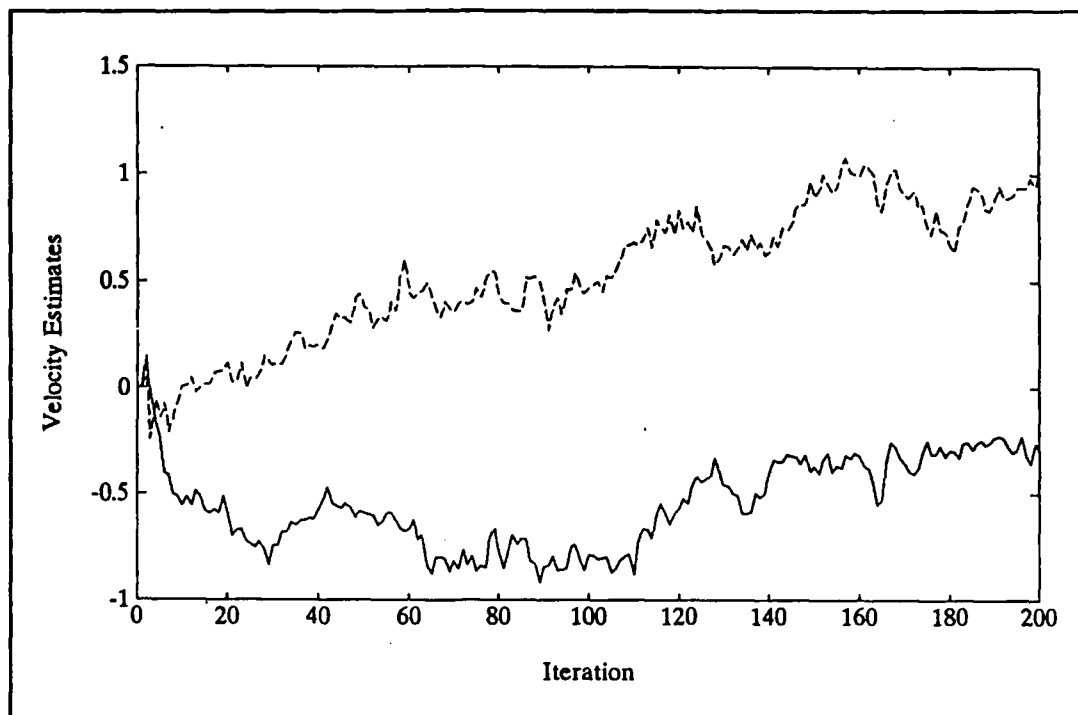


Figure 4.58 Velocity Estimates for Simulation #3.

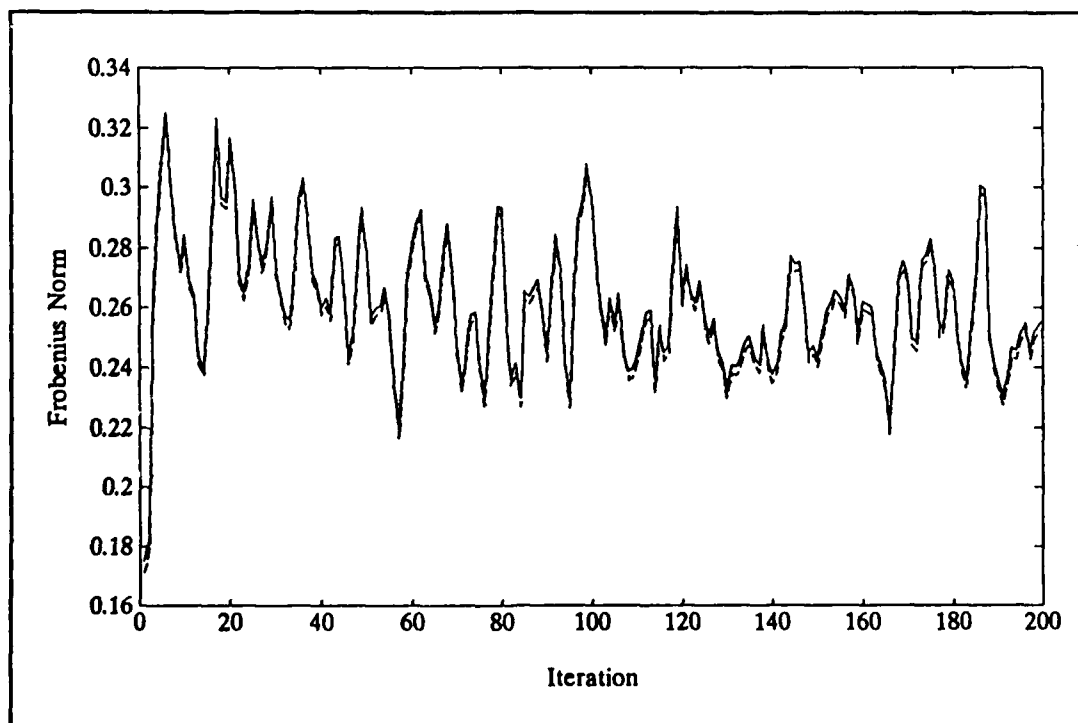


Figure 4.59 Frobenius Norm Results for Simulation #3.

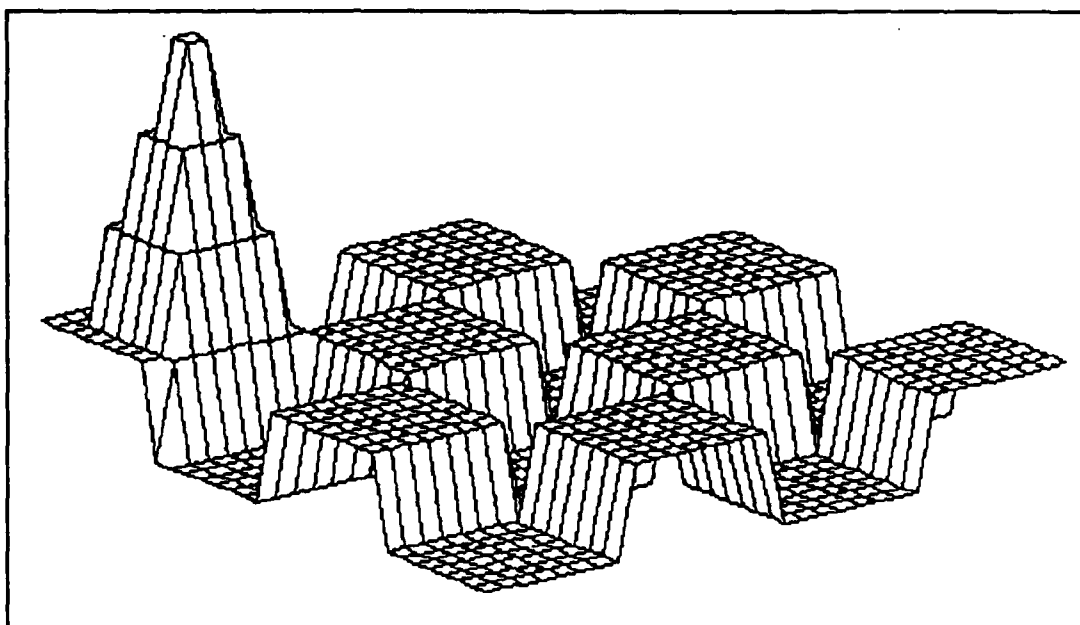


Figure 4.60 Image Frame for Simulation #4 after 65 Iterations.

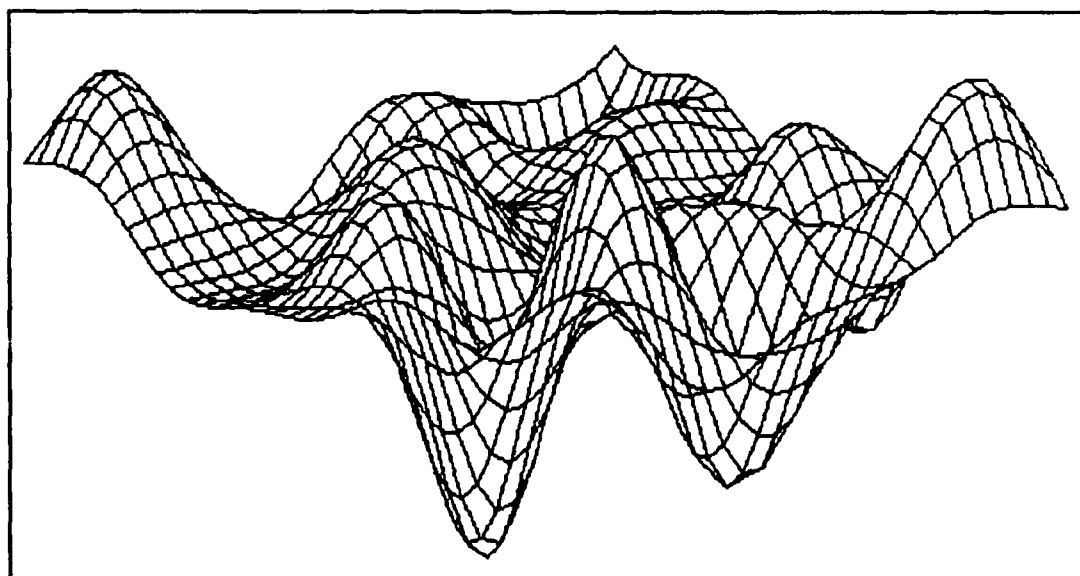


Figure 4.61 Extended Kalman Filter Estimate of Image Frame for Simulation #4 after 65 Iterations.

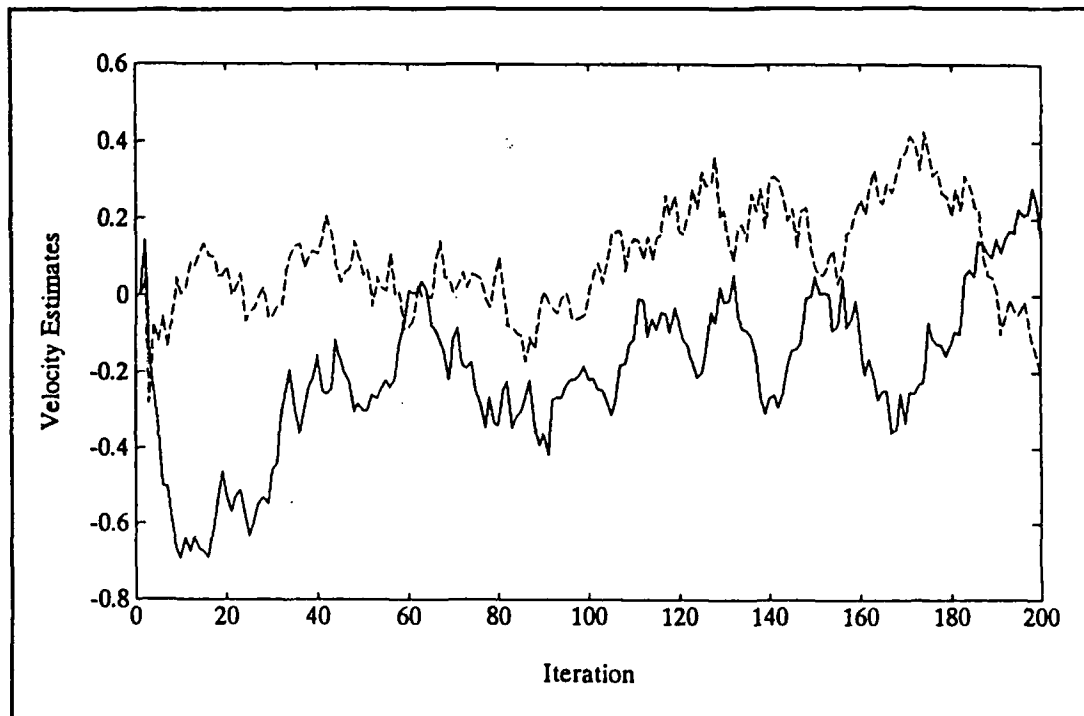


Figure 4.62 Velocity Estimates for Simulation #4.

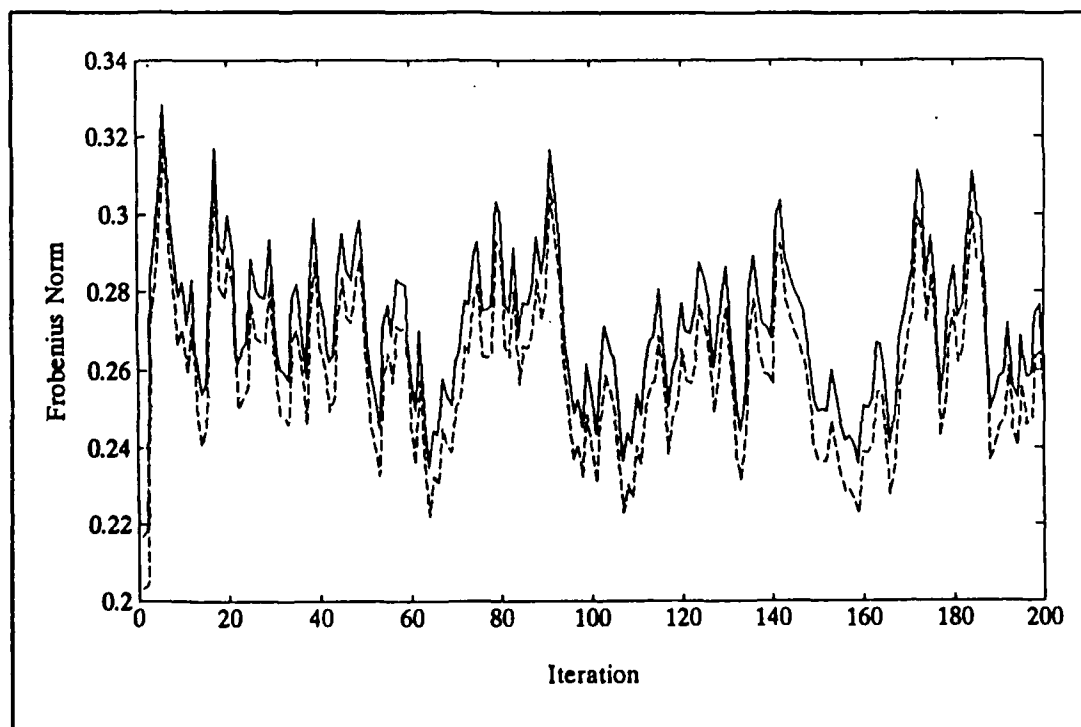


Figure 4.63 Frobenius Norm Results for Simulation #4.

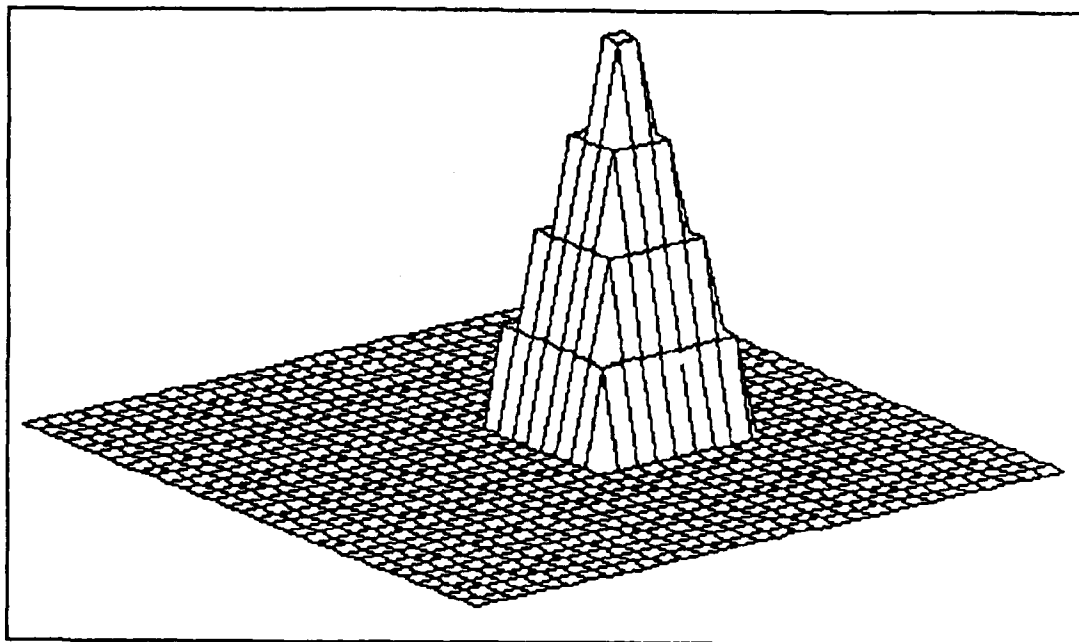


Figure 4.64 Image Frame for Simulation #5 after 95 Iterations.

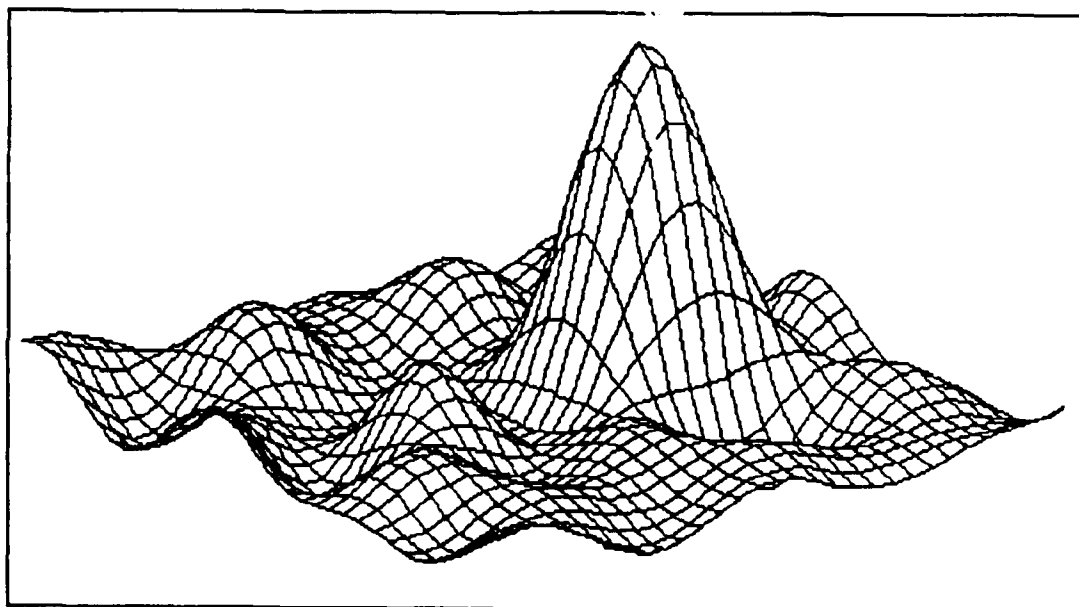


Figure 4.65 Extended Kalman Filter Estimate of Image Frame for Simulation #5 after 95 Iterations.

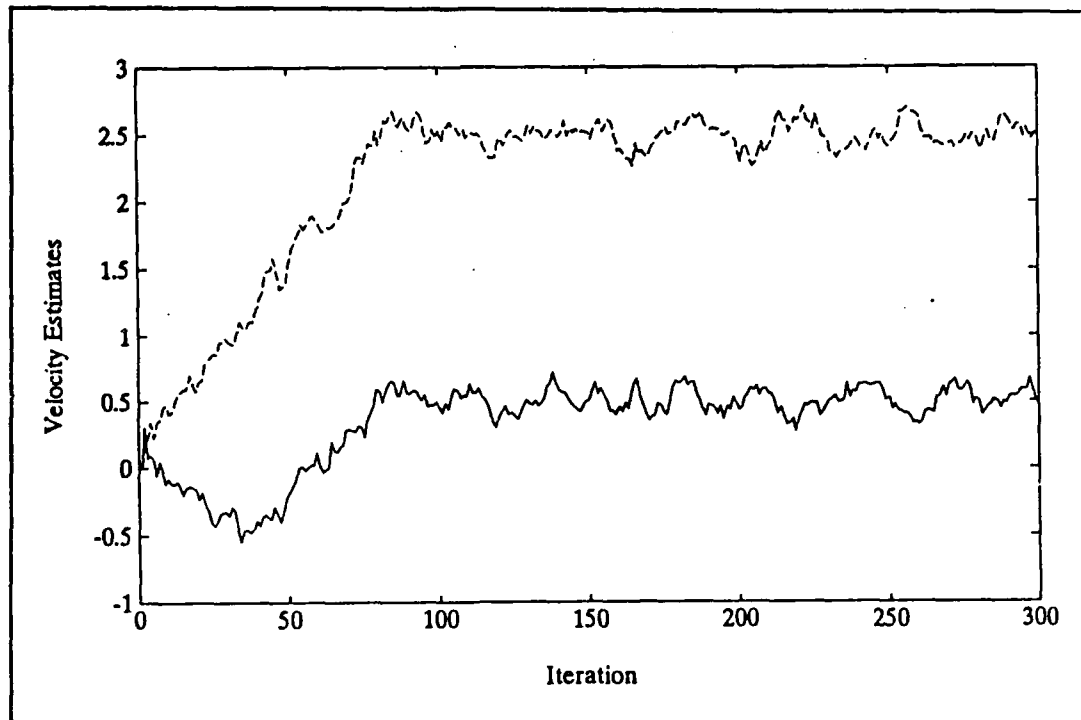


Figure 4.66 Velocity Estimates for Simulation #5.

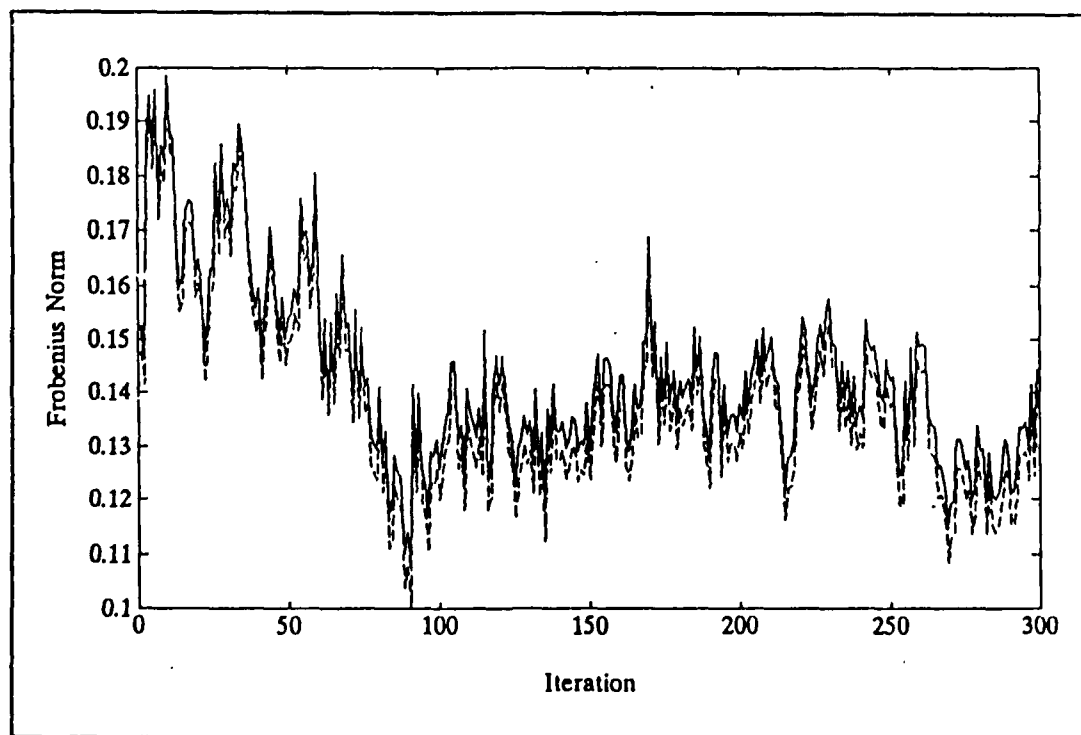


Figure 4.67 Frobenius Norm Results for Simulation #5.

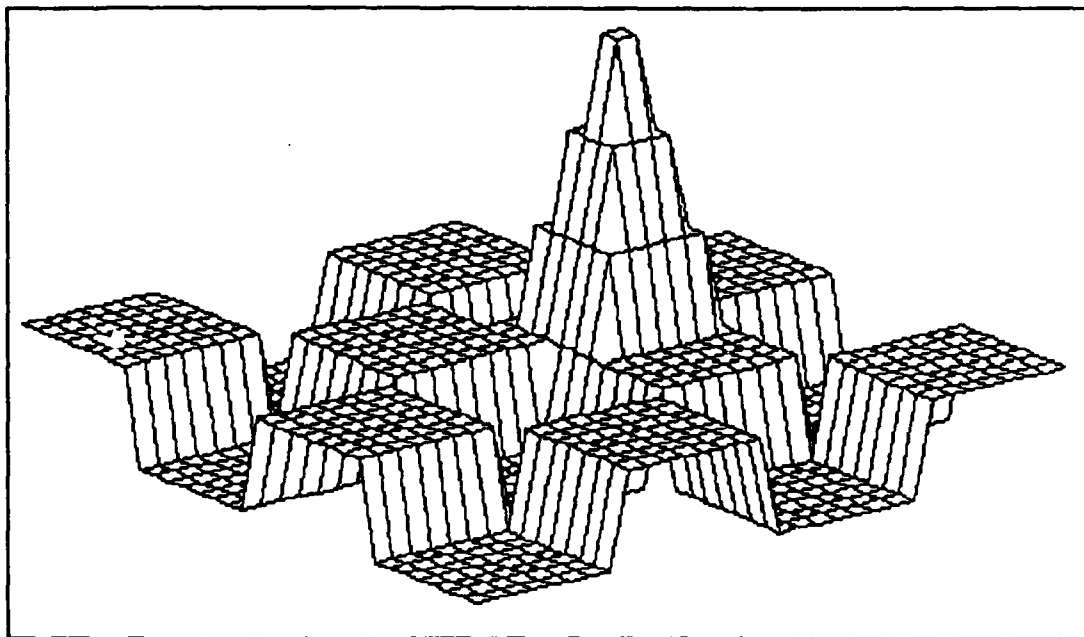


Figure 4.68 Image Frame for Simulation #6 after 95 Iterations.

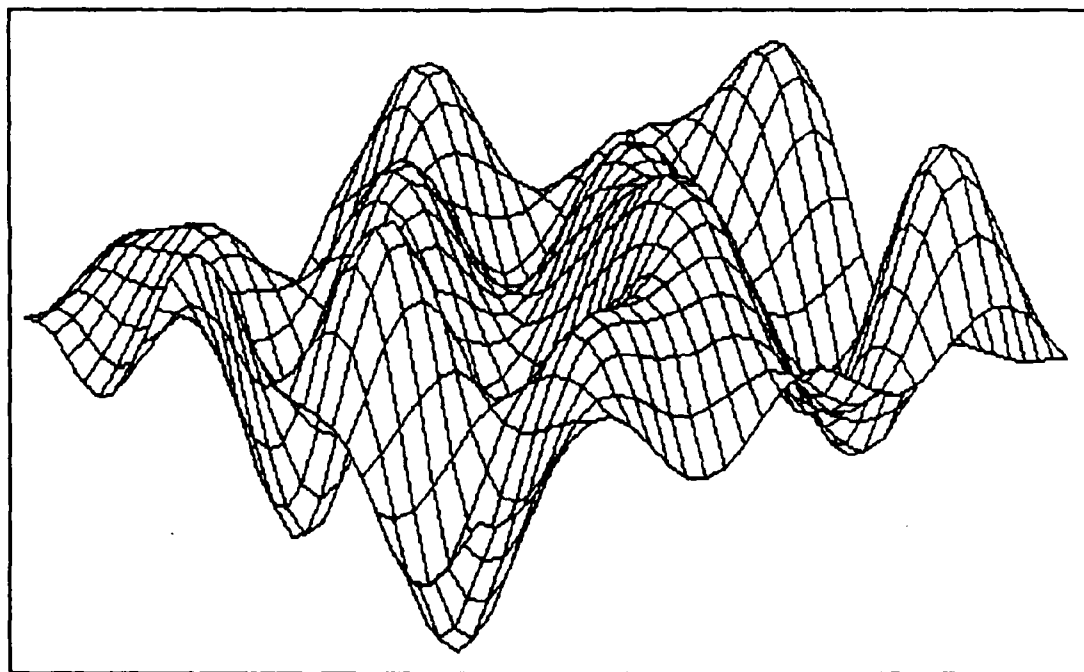


Figure 4.69 Extended Kalman Filter Estimate of Current Image Frame for Simulation #6.

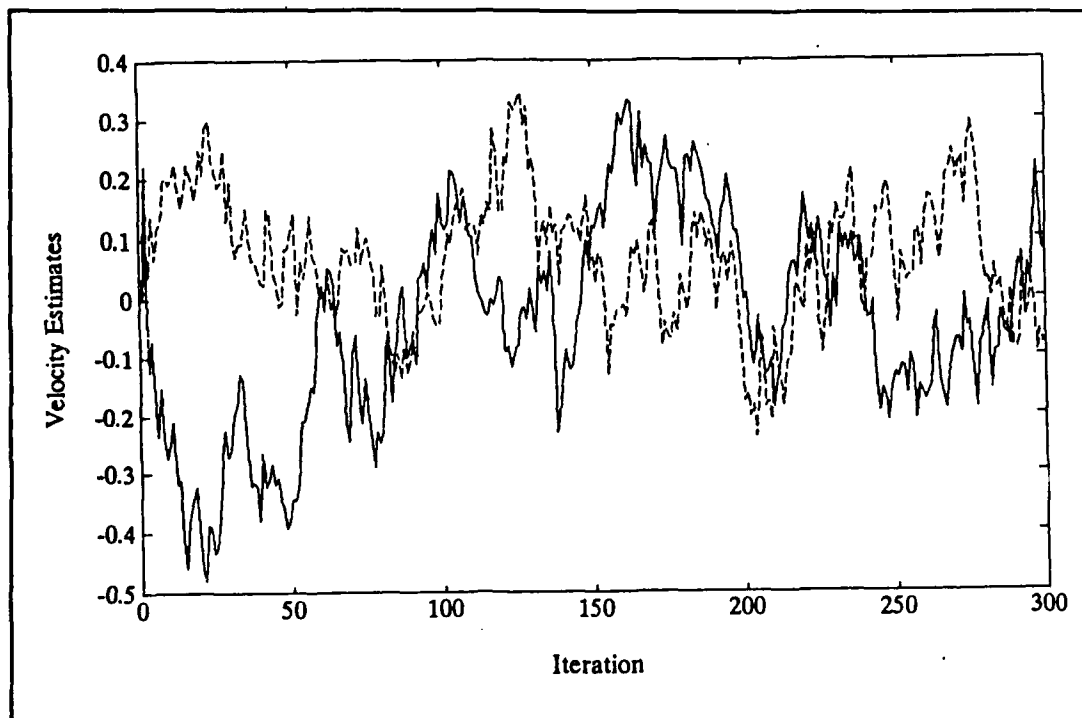


Figure 4.70 Velocity Estimates for Simulation #6.

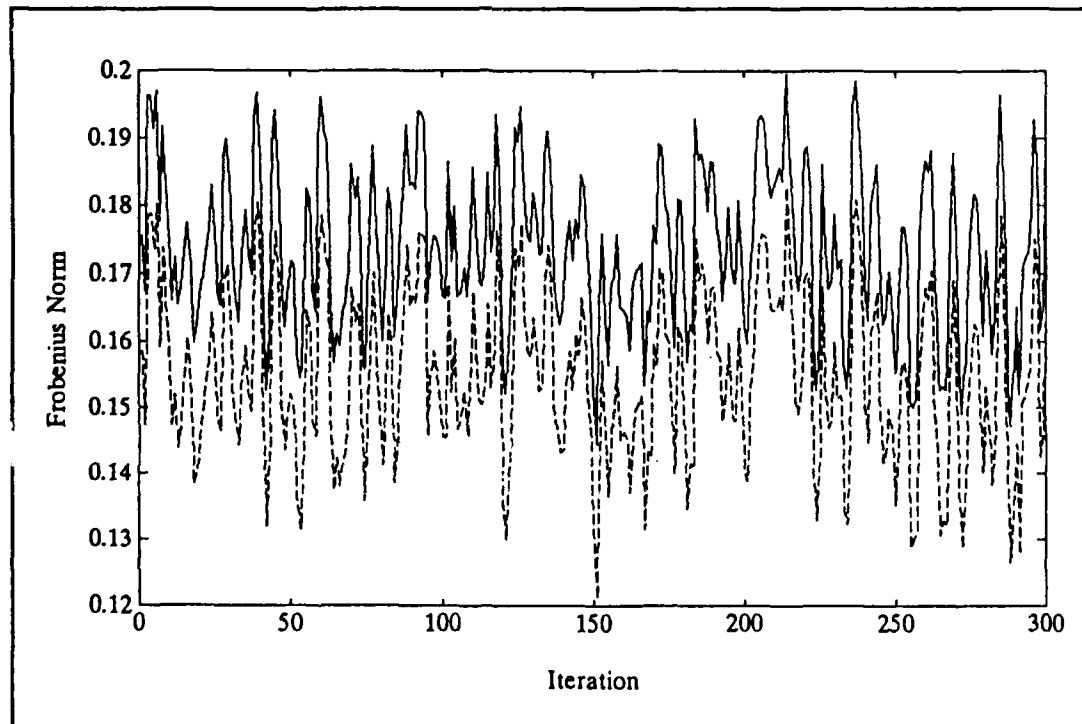


Figure 4.71 Frobenius Norm Results for Simulation #6.

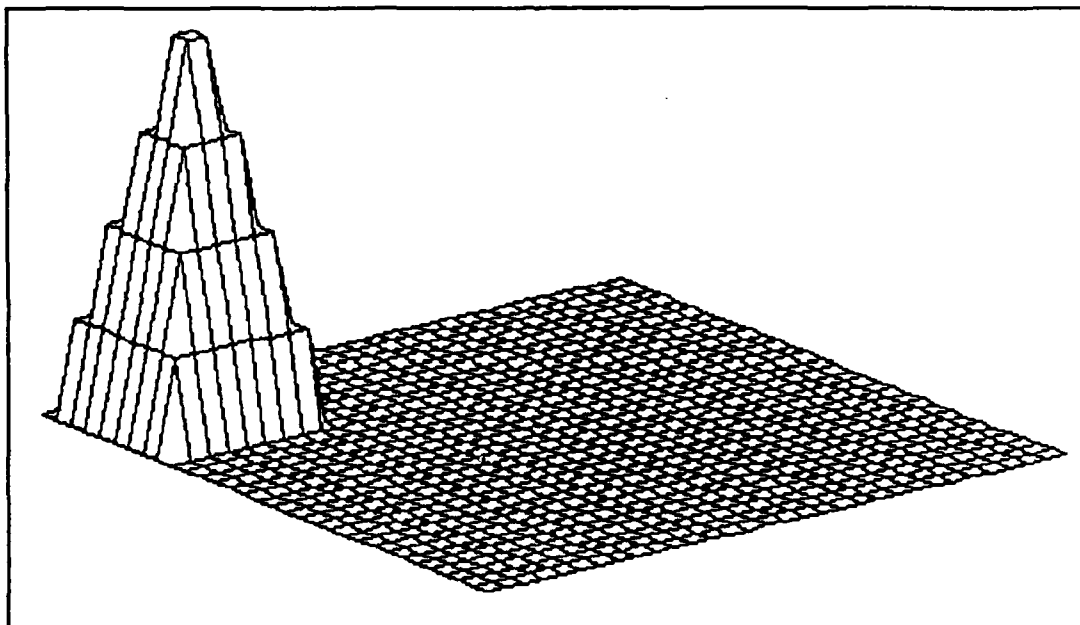


Figure 4.72 Image Frame for Simulation #7 after 65 Iterations.

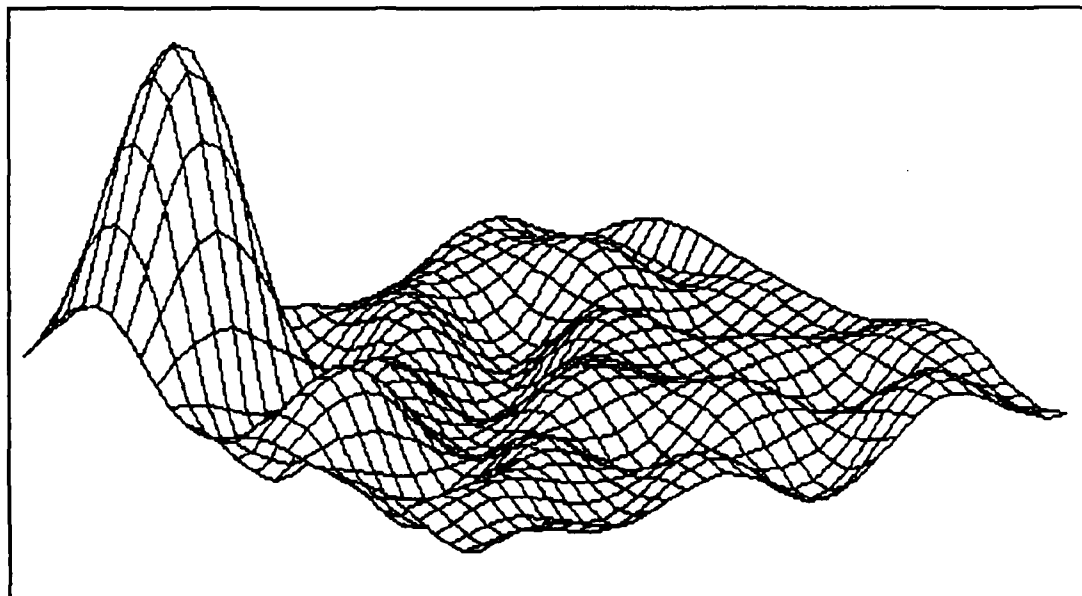


Figure 4.73 Extended Kalman Filter Estimate of Image Frame for Simulation #7 after 65 Iterations.

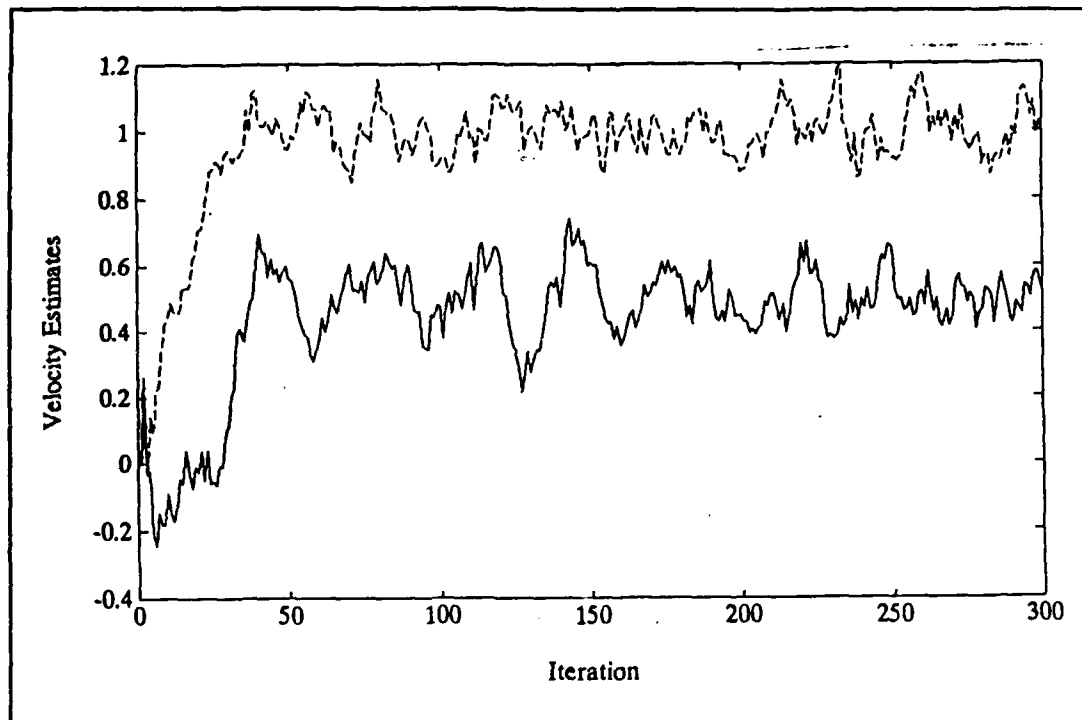


Figure 4.74 Velocity Estimates for Simulation #7.

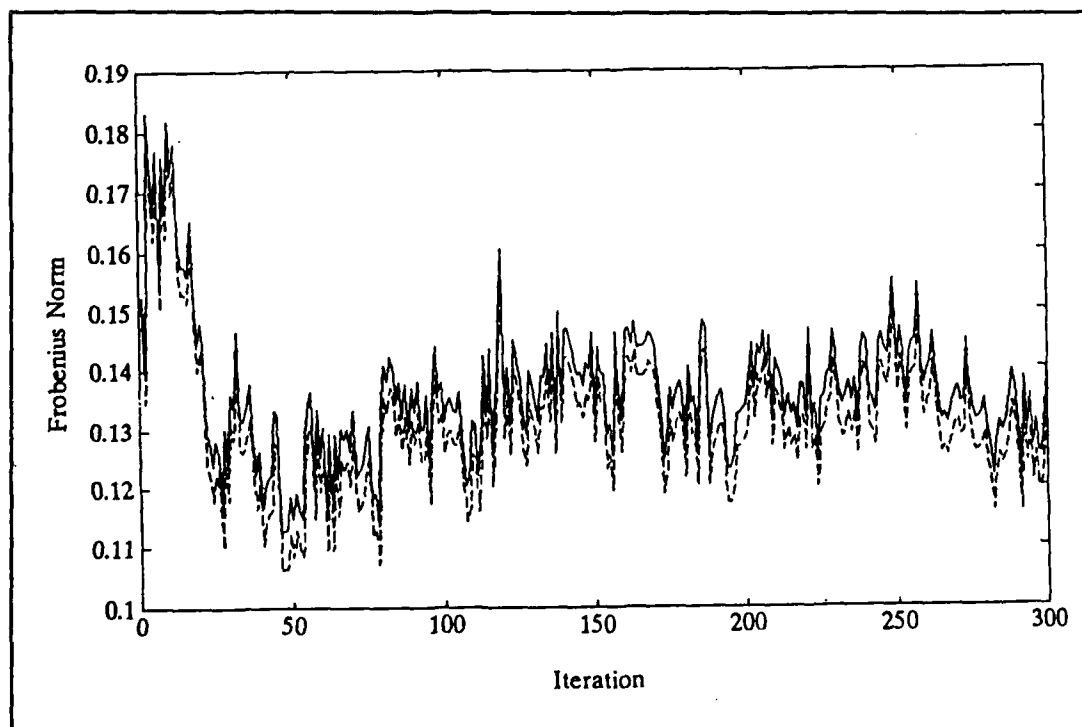


Figure 4.75 Frobenius Norm Results for Simulation #7.

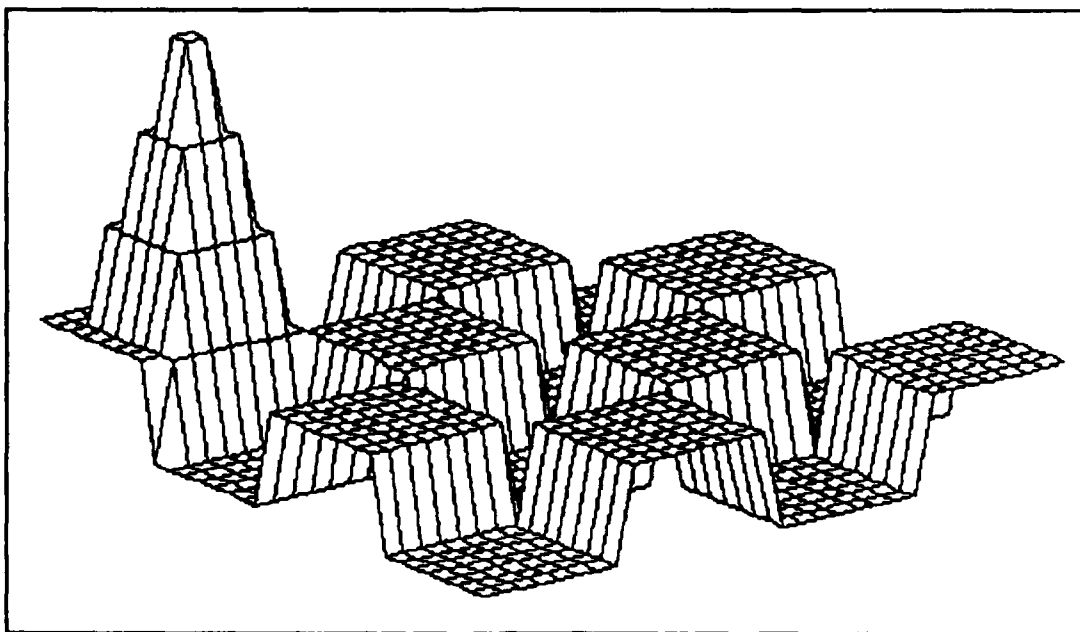


Figure 4.76 Image Frame for Simulation #8 after 65 Iterations.

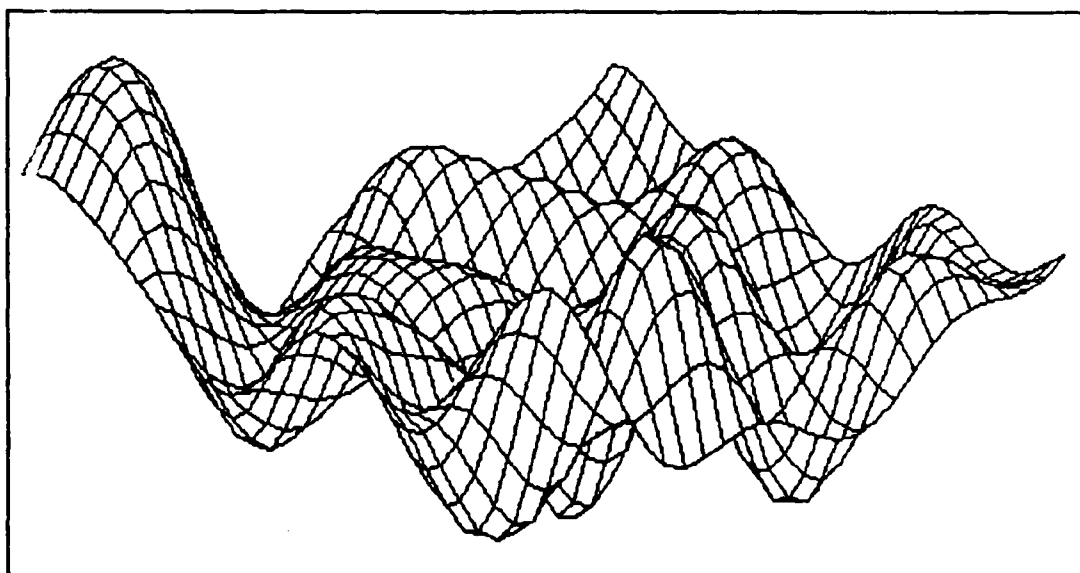


Figure 4.77 Extended Kalman Filter Estimate of Image Frame for Simulation #8 after 65 Iterations.

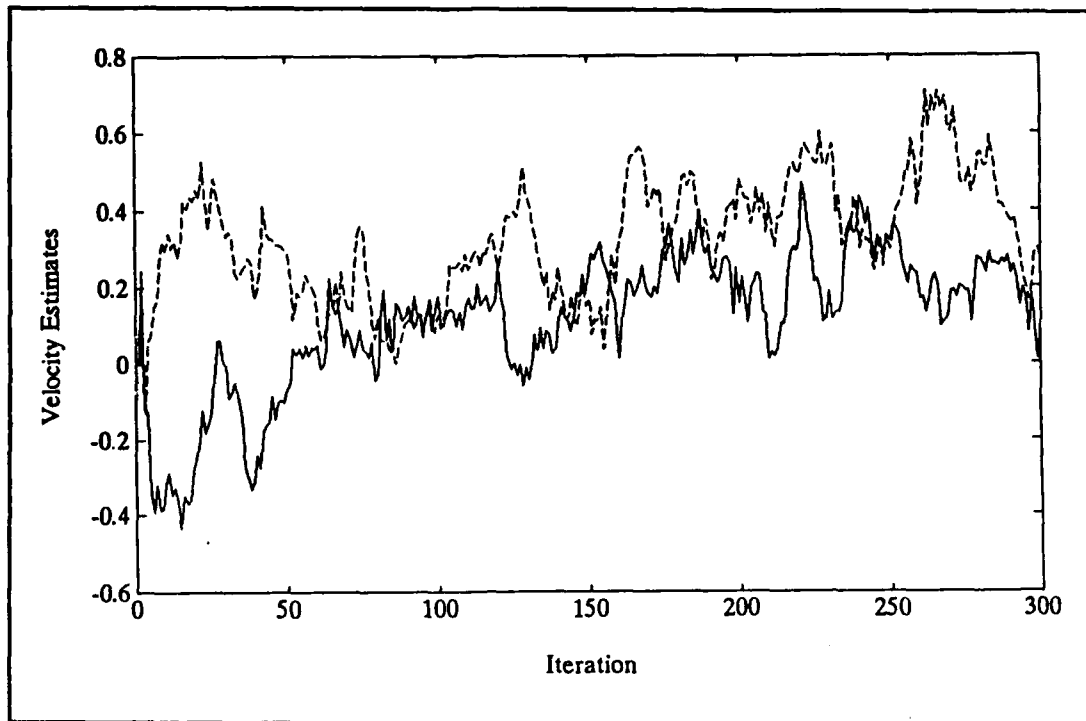


Figure 4.78 Velocity Estimates for Simulation #8.

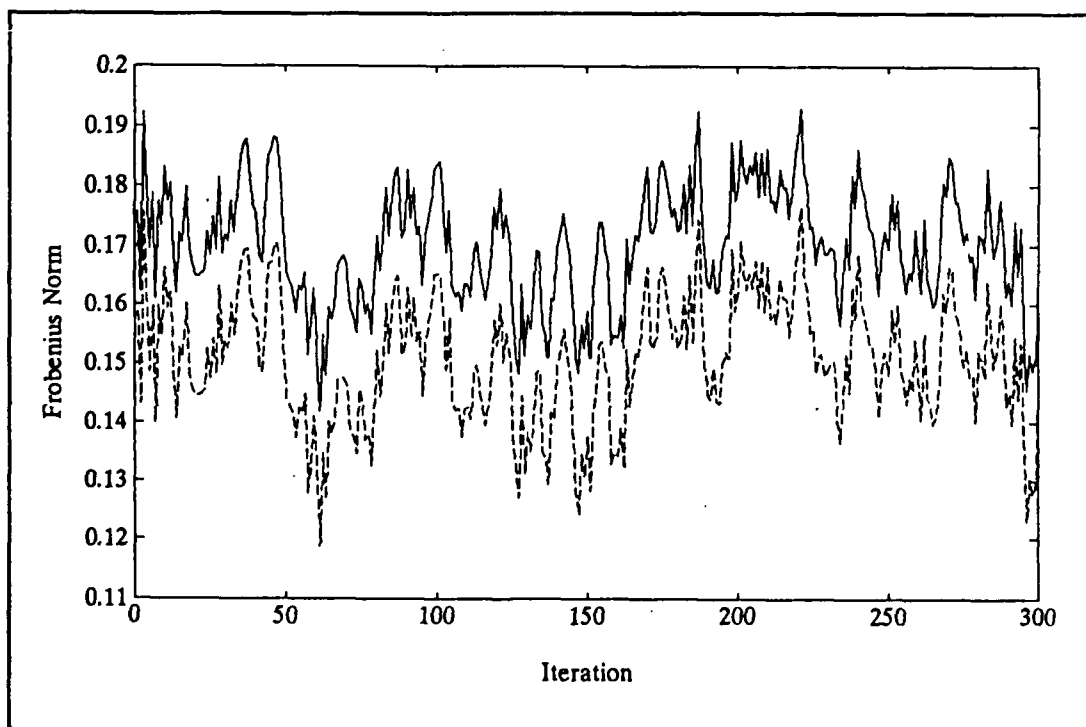


Figure 4.79 Frobenius Norm Results for Simulation #8.

V. CONCLUSIONS

The extended Kalman filter (EKF), as applied to the moving object model, was evaluated under a variety of test conditions. These test conditions included various noise levels, two types of backgrounds, and integer and noninteger velocities. Inclusion of the Frobenius norm provided an analytical measure of the EKF's performance. The performance of the EKF was directly linked with its ability to detect and track the test object's motion.

For simulations using a homogeneous background, the EKF was unable to detect and track the square test object when the standard deviation of the noise was raised to 4.0. Similar results occurred when the pyramid test object was used and the standard deviation was increased to 2.0.

The detection of the test object by the EKF is also related to the signal energy content of the test object in relation to the energy contained in the image frame background. The addition of the checkerboard background increased the energy content of the background. As a result, the simulations showed that the detection threshold of the EKF was reduced. The EKF was precluded from detecting and tracking the square test object's motion across a checkerboard background when the standard deviation of the noise was raised above 2.0. Even with a standard deviation of 1.0, the EKF was unable to detect and track the pyramid test object.

These computer simulations have demonstrated that the EKF algorithm can successfully operate in a high-noise environment. These results are encouraging and warrant continued research.

APPENDIX A. TUTORIAL ON TWO-DIMENSIONAL SPATIAL FREQUENCIES

A. PROGRAM DISCUSSION

The program SPAPICT.M begins by placing an 8x8 square object of unity amplitude in the center of the frame (Fig. A.1). The image frame is then transformed to the frequency domain where each spatial frequency component is defined by the discrete Fourier transform [Ref. 2].

$$F(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1, n_2) \exp \left[-j2\pi \left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2} \right) \right], \quad (A.1)$$

where $0 \leq k_1 \leq N_1-1$ and $0 \leq k_2 \leq N_2-1$.

The transformed image frame is sent through an ideal lowpass filter (LPF), i.e., the high frequency components are truncated and set equal to zero. The filtered image frame is then transformed back to the spatial domain where the sharp edges are lost due to the high frequency components being filtered out (Fig. A.2).

The image frame is transformed into a matrix that exhibits conjugate symmetry. Conjugate symmetry results when a real image is transformed into the frequency domain. The number of computations for each spatial frequency and its complex conjugate can be reduced using Euler's identity,

$$e^{j\omega} + e^{-j\omega} = 2 \cos(\omega) , \quad (A.2)$$

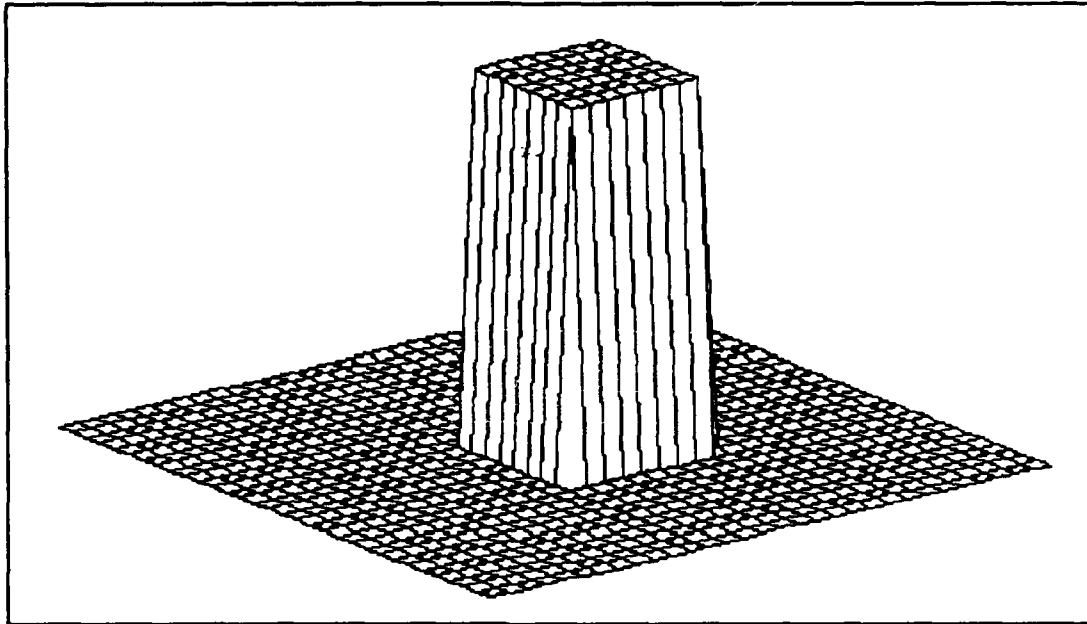


Figure A.1 An 8x8 Square Image of Unity Amplitude Depicted on a 32x32 Pixel Frame.

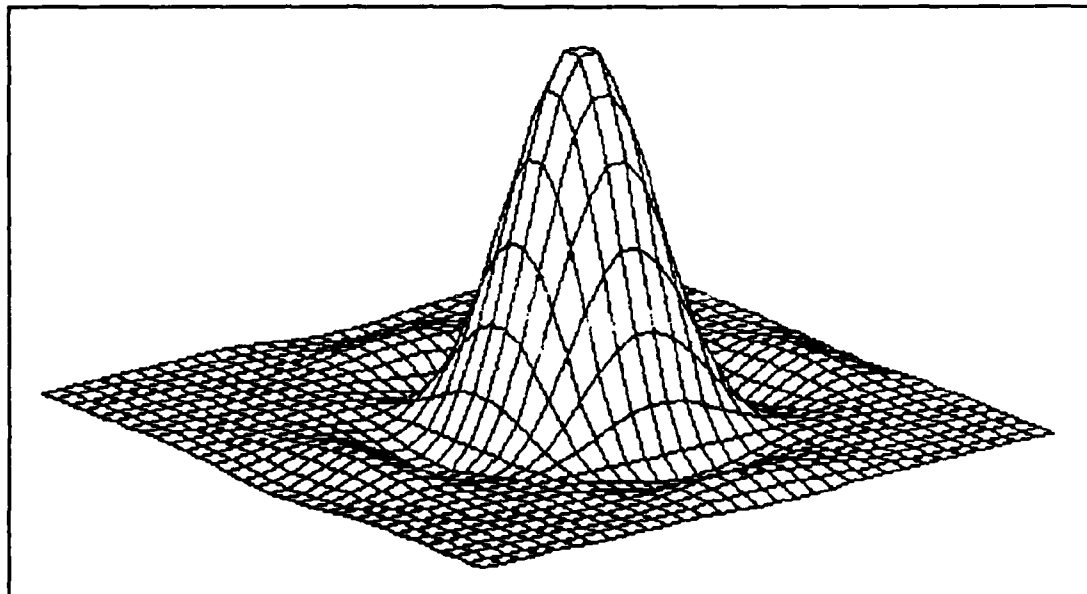


Figure A.2 Filtered Frame after being Transformed Back to the Spatial Domain.

where $\omega = k_1 n_1 + k_2 n_2$ for the two-dimensional case. The contribution of each spatial frequency and its complex conjugate was computed using

$$F(k_1, k_2) + F^*(k_1, k_2) = 2 |F(k_1, k_2)| \cos \left[\frac{2\pi}{N} (n_1 k_1 + n_2 k_2) + \angle F(k_1, k_2) \right], \quad (A.3)$$

where $N = 32$.

As an example, the spatial frequency $F(-1, -1)$ is the complex conjugate of the spatial frequency $F(1, 1)$. In the reconstruction of the filtered frame in the spatial frequency domain, the contribution of a specific spatial frequency and its complex conjugate were calculated using Eqn. A.3 and then summed with the previous frame.

Figures A.3-A.8 show the successive summation of each spatial frequency and its corresponding complex conjugate in the reconstruction of the filtered frame in the spatial frequency domain. Each figure depicting a specific spatial frequency has been scaled and phase shifted based on the Fourier coefficients that were computed using the DFT.

Figure A.3 depicts the summation of the dc component and $F(0, 1)$ and its complex conjugate. Figure A.4 depicts the contribution of spatial frequencies $F(0, 2)$ and $F^*(0, 2)$ which are then summed with the frame shown in Fig. A.3 to produce Fig. A.5.

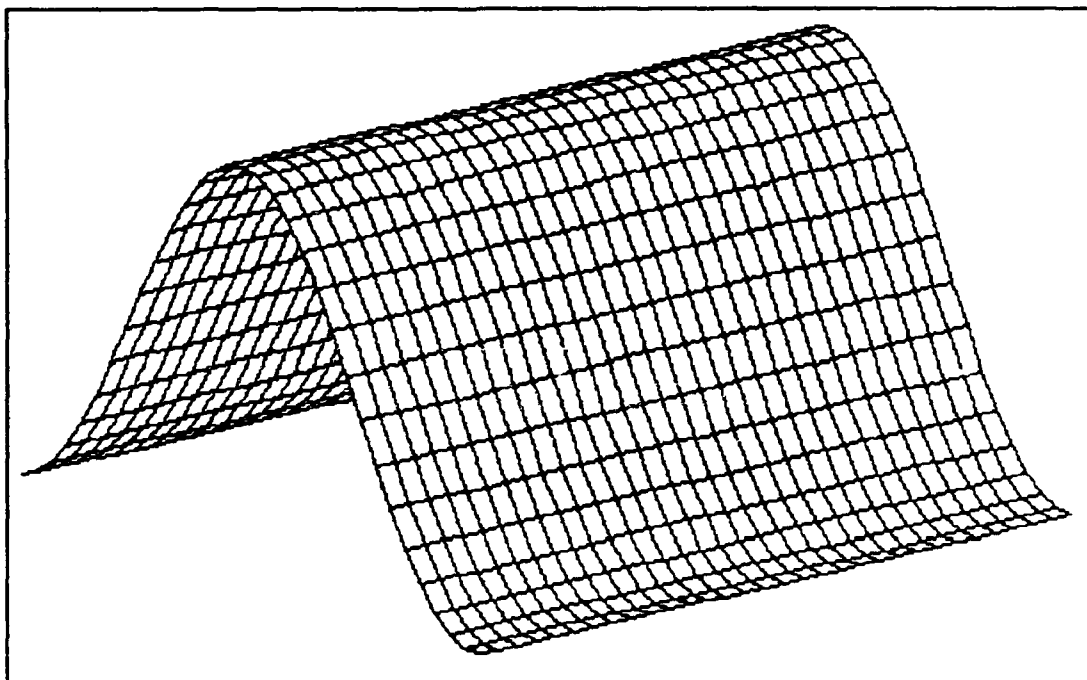


Figure A.3 Summation of the dc component, $F(0,1)$, and $F^*(0,1)$.

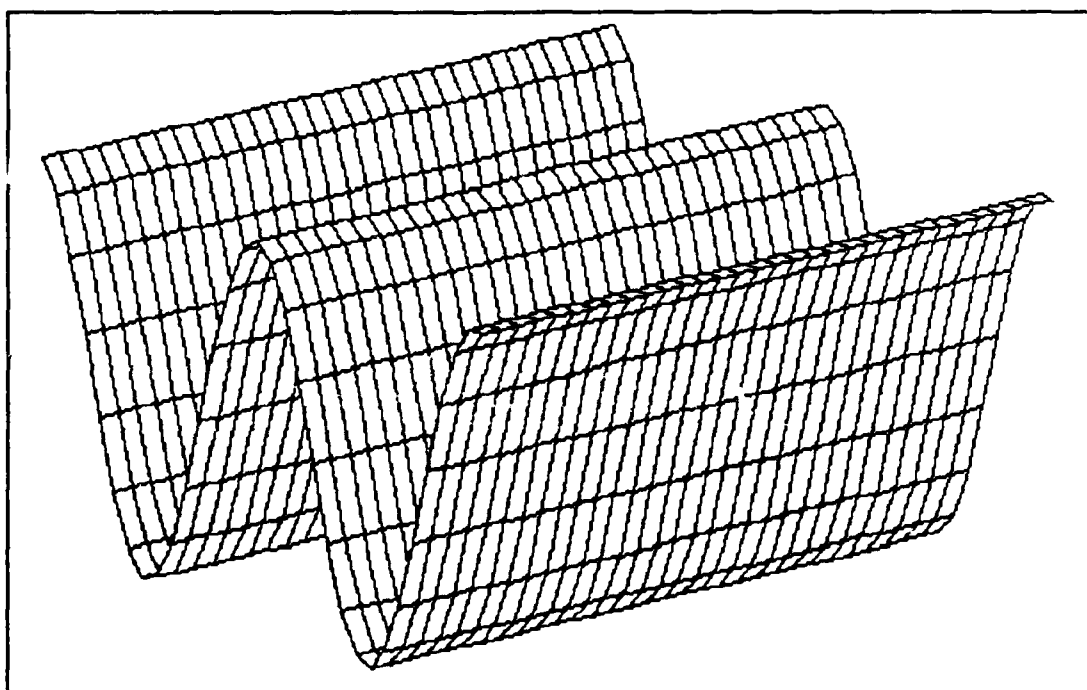


Figure A.4 Spatial Frequency $F(0,2)$

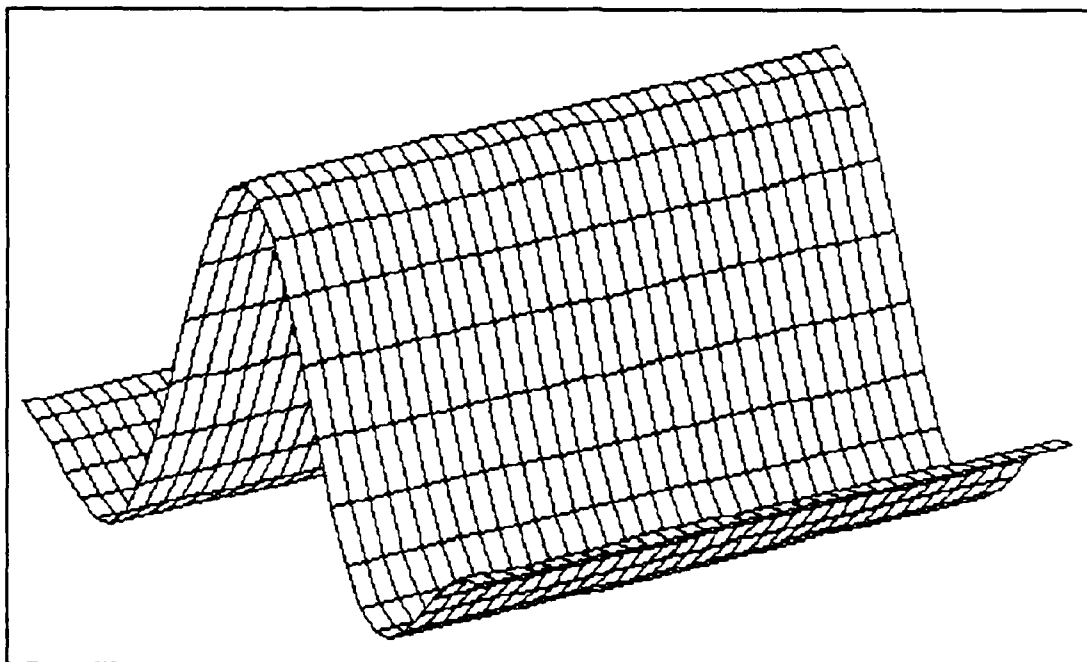


Figure A.5 Summation of Previous Spectral Frequencies, $F(0,2)$, and $F^*(0,2)$.

This process is continued with the contributions from the other spatial frequencies being computed and summed with the frame shown in Fig. A.5. Figure A.6 shows the spatial frequency $F(2,1)$ which has two cycles in the n_1 -direction and one cycle in the n_2 -direction. Figure A.7 shows the summation of all the contributions up to this point. The final frame shown in Fig. 8 is the same as the frame depicted in Fig. A.2.

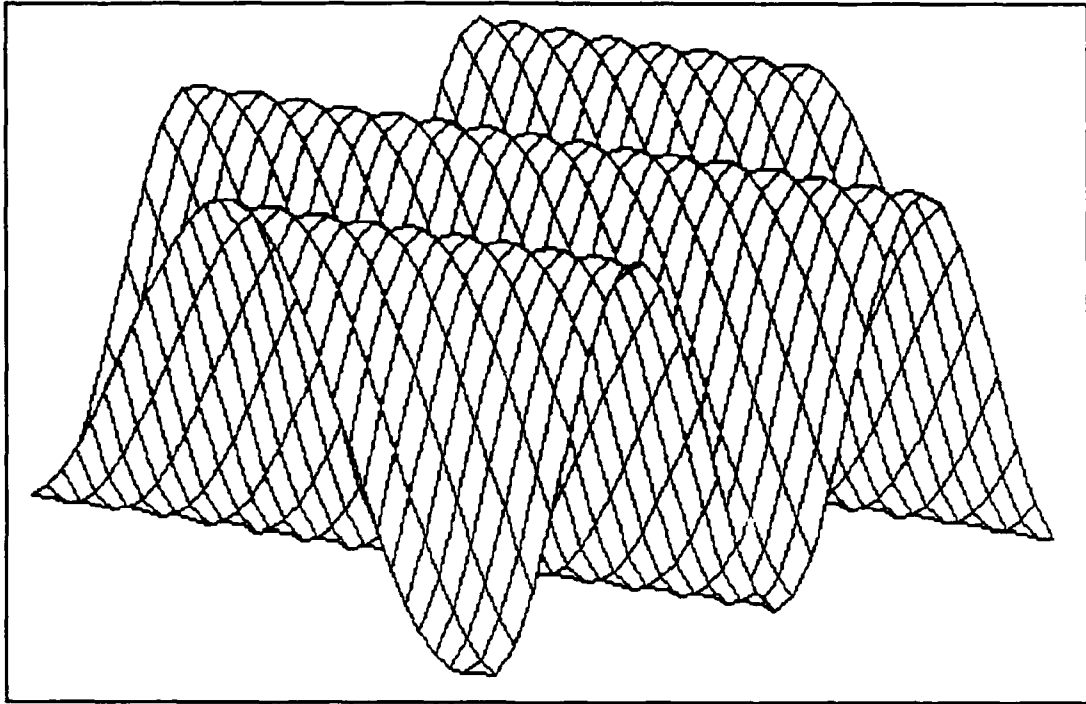


Figure A.6 Spatial Frequency $F(2,1)$

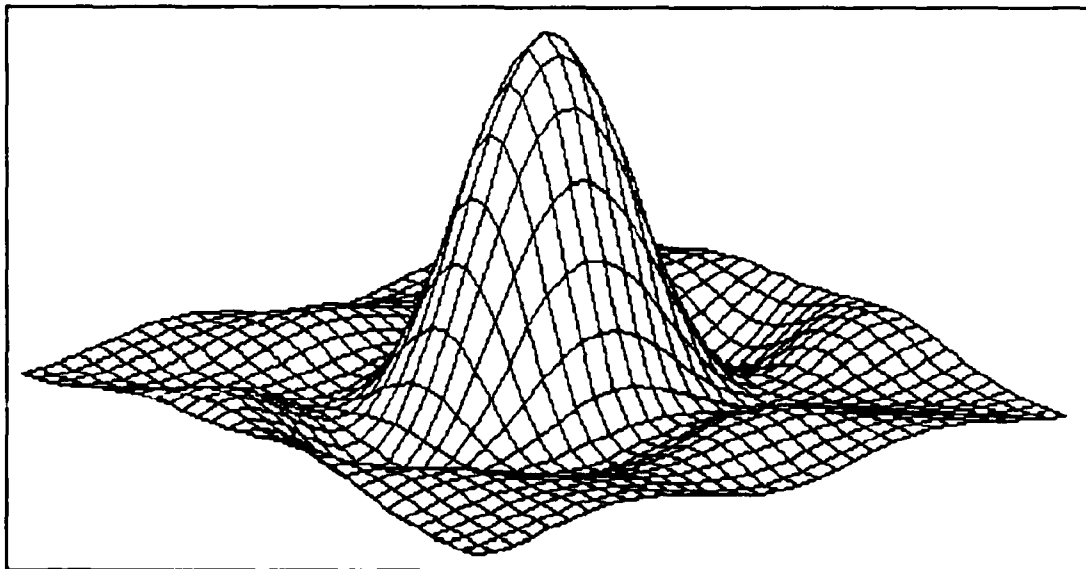


Figure A.7 Summation of Previous Spectral Frequencies and $F(2,1)$ and $F^*(2,1)$.

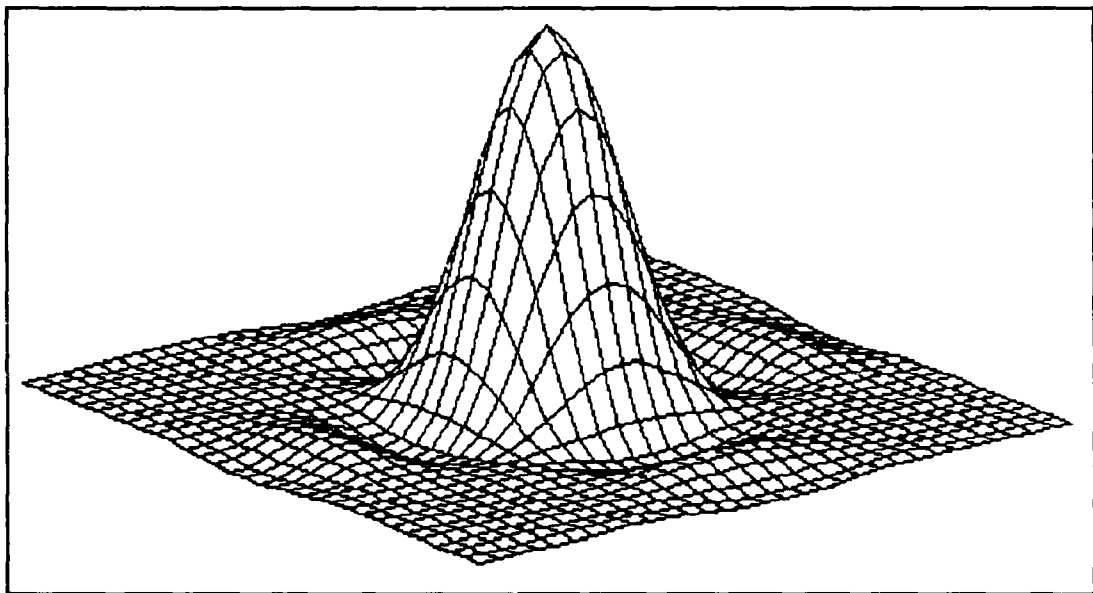


Figure A.8 Final summation of all spectral frequencies.

B. PROGRAM LISTING

SPAPIC.M is a stand-alone MATLAB program that reconstructs a filtered image in the frequency domain.

```
*****
% VARIABLE DEFINITION
%
% pict      : Frame with square image as it would appear in
%             spatial domain
% img       : 8x8 square image with unity amplitude
% freq      : Frequency vector listing the pertinent
%             spatial frequencies needed to
%             reconstruct the filtered image in the
%             spatial frequency domain
% fftpic    : Frame after transformation to the spatial
%             frequency domain
% freqpict  : Frame after lowpass filtering in spatial
%             frequency domain
% pictifft  : Lowpass filtered Frame after transformation
%             back to spatial domain
%
%*****

% CREATE THE PICTURE

img = ones(8);
pict1 = zeros(32);
pict1([14:21],[14:21]) = img;
[ n1 n2 ] = size(pict1);

% CREATE FREQUENCY VECTOR

n = 0;
for i = 0:3,
    if i == 0,
        for j = 0:3,
            n = n + 1;
            freq(n,:) = [ (2*pi*i)/n1 (2*pi*j)/n2 ];
        end
    else
        for j = -3:3,
            n = n + 1;
            freq(n,:) = [ (2*pi*i)/n1 (2*pi*j)/n2 ];
        end
    end
end
end
```

```

% PLOT THE FRAME IN THE SPATIAL DOMAIN

mesh(pict1)
title('Frame -- Spatial Domain')
meta spaplot

% TAKE FOURIER TRANSFORM OF FRAME

fftpict1 = fftshift(fft2(pict1));

% PASS THE TRANSFORMED FRAME THROUGH A LOWPASS FILTER

trnpict = fftpict1([14:20],[14:20]);
trnpict1 = trnpict(:);

% TAKE INVERSE FOURIER TRANSFORM OF FILTERED FRAME

freqpict = zeros(32);
freqpict([14:20],[14:20])= trnpict;
pictifft = ifft2(fftshift(freqpict));

% PLOT THE FILTERED FRAME IN THE SPATIAL DOMAIN

mesh(pictifft)
title('Lowpass Filtered frame in the spatial domain')
meta spaplot
pause(5)

% DISPLAY AND SAVE THE MAGNITUDE AND PHASE OF THE %
TRANSFORMED PICTURE

diary spadiary.mat
disp('Lowpass Filtered Frame -- Magnitude and Phase')
z1 = [ abs(trnpict1'); angle(trnpict1') ]
diary off

% CONSTRUCT THE FRAME IN THE FREQUENCY DOMAIN USING
% FREQUENCIES THAT WERE NOT TRUNCATED

z1 = z1(:,1:25);
z1 = flipud(z1');
pict = zeros(32);

for n = 1:25,
    u    = freq(n,1);
    v    = freq(n,2);
    mag   = z1(n,1);
    phase = z1(n,2);
    x     = 0:31;
    y     = 0:31;
    [ xx,yy ] = meshdom(x,y);
    z      = 2 * mag * cos(u*xx + v*yy + phase);

```

```

    pict = pict + z;
    subplot(211), mesh(z)
    title('Constructing truncated image in frequency '...
          'domain')
    xlabel(['f(', num2str(u), ', ', num2str(v), ')'])

    subplot(212), mesh(pict)
    xlabel(['Adding f(', num2str(u), ', ', num2str(v), ') and'...
          ' f*(', num2str(u), ', ', num2str(v), ') to the '...
          'picture'])
    meta spaplot
    pause(3)
    clg
end

clg, subplot(111)

% GET A PLOT OF THE FINAL IMAGE CONSTRUCTED IN THE FREQUENCY
% DOMAIN

mesh(pict)
title('Final image constructed in the frequency domain')
meta spaplot

```


APPENDIX B. PPHSE.M

A. PROGRAM DISCUSSION

PPHSE.M is a stand-alone program written for use with MATLAB to demonstrate the spatial frequency phase shift that results from moving an object across an image frame in the spatial domain.

PPHSE.M sequentially moves an 8x8 unity square across a 32x32 image frame in the spatial domain for 50 iterations. Two image frames are transformed to the spatial frequency domain. The first image frame consists of a homogeneous background with no added noise. The second image frame has zero mean, Gaussian white noise.

The real and imaginary Fourier coefficients and the phase in degrees are recorded and plotted for both the no-noise and noise-corrupted image frames for the two spatial frequencies $F(1,0)$ and $F(0,1)$. Only the phase is worth noting since the magnitude of the transformed image frame remains constant. In the spatial frequency domain, the effect of moving the square object across the image frame is a phase shift in each of the spatial frequency components.

Figure B.1 shows the phase shift in degrees for the spatial frequency $F(1,0)$. The effect of moving the square object across the image frame 1 pixel per iteration is a 11.25° phase shift per iteration. The dashed line represents the noise-

corrupted image frame. The phase shift is no longer perfectly linear, but it still tracks the no-noise phase shift rather closely.

Figure B.2 shows the phase shift in degrees for the spatial frequency $F(0,1)$. The effect of simultaneously moving the square object 2 pixels per iteration vertically is a 22.5° phase shift per iteration. The phase shift for the noise corrupted image frame is represented by the dashed line and the nonlinear phase shift again generally follows the no-noise phase shift.

It should be noted that the phase shift due to the object motion is a function of the spatial frequency. In this example, $F(1,0)$ and $F(0,1)$ were chosen because they represent the spatial frequencies that purely describe the horizontal and vertical motion of the moving object.

Figures B.3-B.6 sequentially show the magnitude of the real and imaginary Fourier coefficients for the spatial frequencies $F(1,0)$ and $F(0,1)$. The dashed line again represents the coefficients of the noise-corrupted image frame.

The real and imaginary Fourier coefficients for each of the spatial frequencies were two of the three the states that were estimated by the extended Kalman filter described in this thesis.

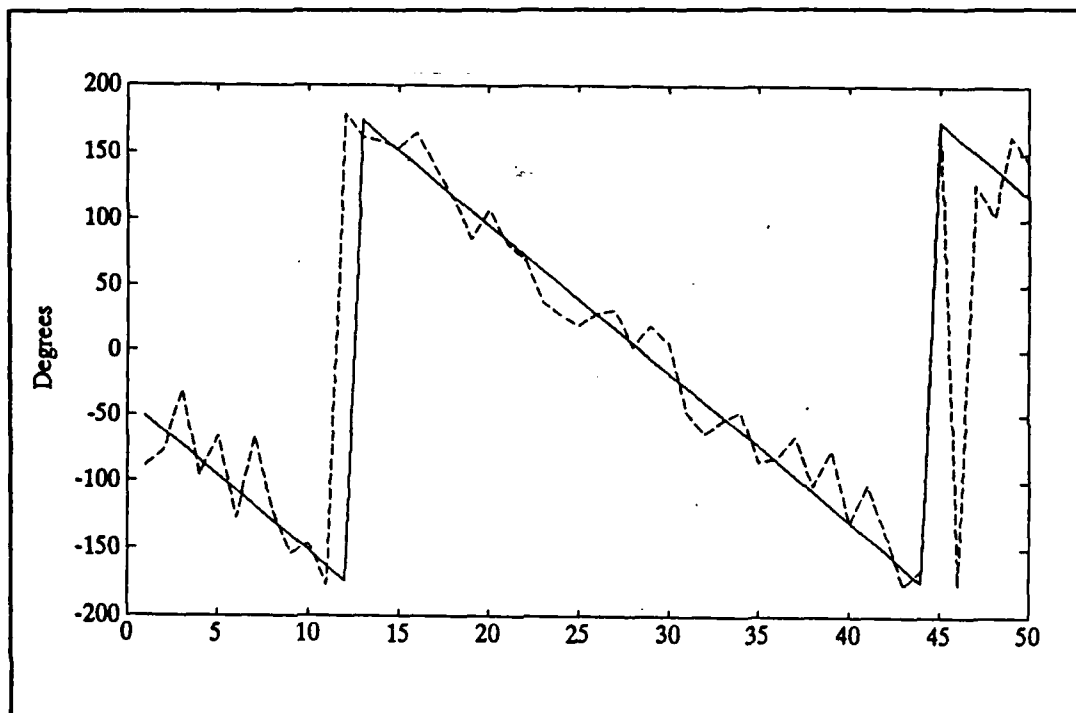


Figure B.1 Phase Shift in Degrees for Spatial Frequency $F(1,0)$.

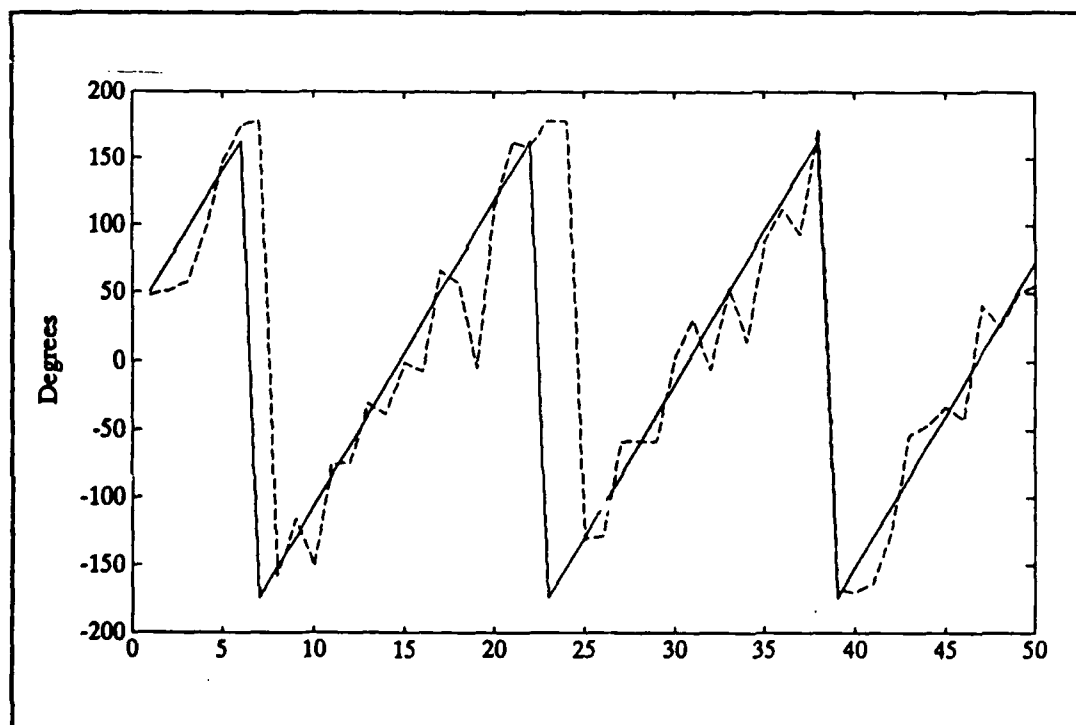


Figure B.2 Phase Shift in Degrees for Spatial Frequency $F(0,1)$.

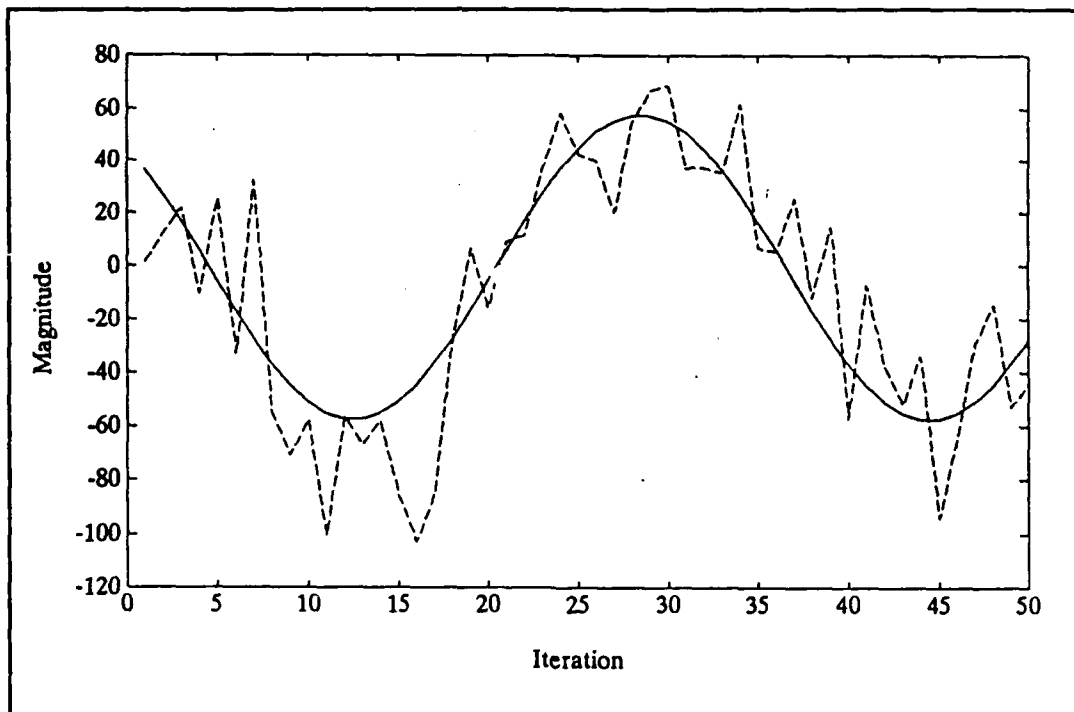


Figure B.3 Real Fourier Coefficients for Spatial Frequency $F(1,0)$.

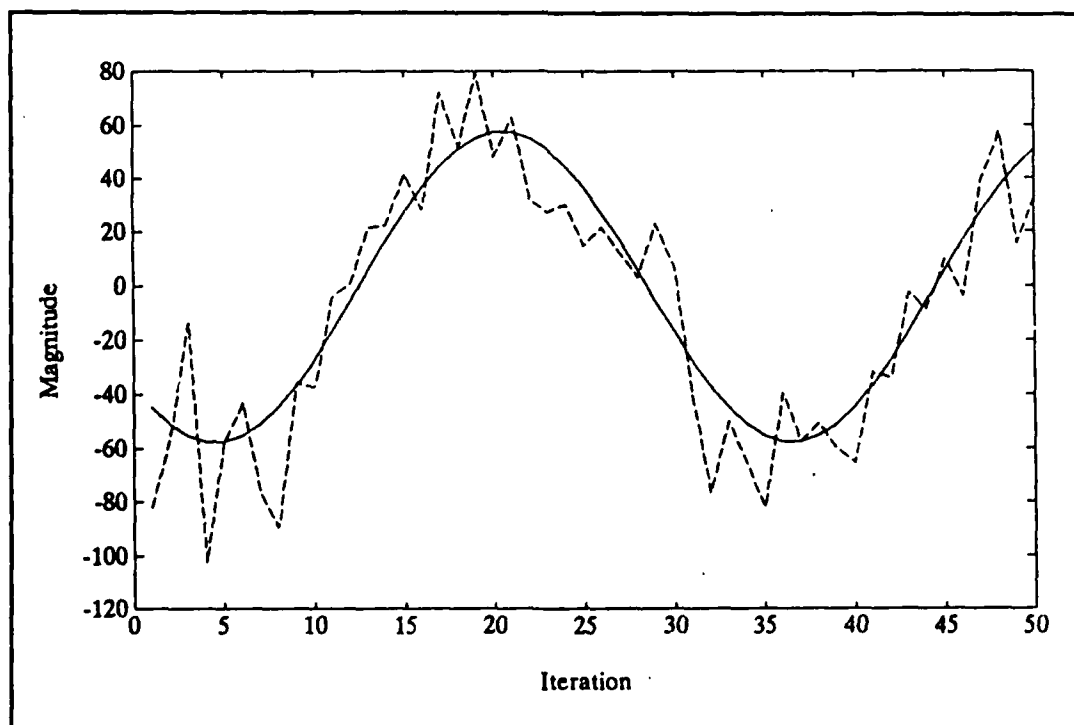


Figure B.4 Imaginary Fourier Coefficients for Spatial Frequency $F(1,0)$.

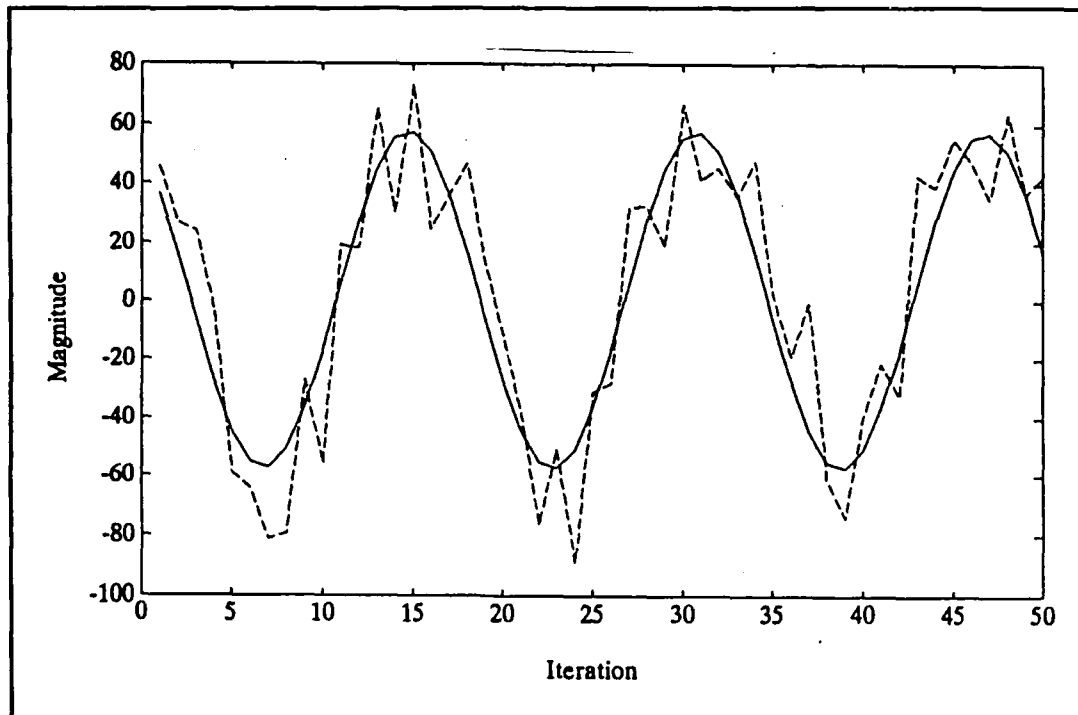


Figure B.5 Real Fourier Coefficients for Spatial Frequency $F(0,1)$.

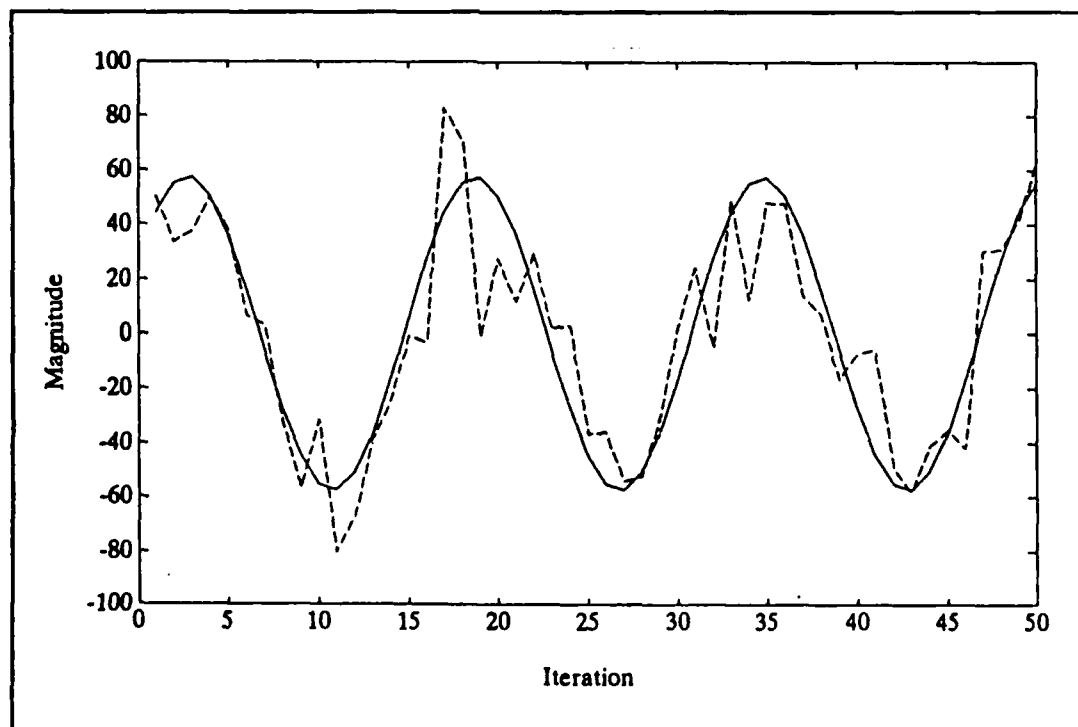


Figure B.6 Imaginary Fourier Coefficients for Spatial Frequency $F(0,1)$.

B. PROGRAM LISTING

```
% PPHSE.M is a stand-alone program that demonstrates
% the spatial frequency phase shift that results when
% a moving object traverses an image frame in the
% spatial domain

clear                % clear out all variables in memory
clear functions

shpe = ['Square']; % Define the shape of the moving image
vrow = 2.0;        % Vertical Velocity -- # of pixels/its
vclm = 1.0;        % Horizontal Velocity -- # of pixels/its
imax = 3;          % Size of the spatial frequency truncation
jmax = 3;
its = 50;          % Number of iterations
sig1 = 1.0;        % Standard deviation of the noise

shape = ones(8);    % Define the shape -- square of %
                    % unity amplitude
pict = zeros(32);   % Create digital array to place %
                    % image in
rand('normal');      % Establish normal distribution
rand('seed',2380849) % Define a standard random seed
[ n1 n2 ] = size(shape); % Define the size of the image

% CREATE STORAGE VECTORS FOR CALULATED DATA

phase0=zeros(2,its); % Phase shift in degrees -- no noise
phase1=zeros(2,its); % Phase shift in degrees -- 1.0 Std dev

rslt0 = zeros(2,its); % Fourier coef. -- no noise
rslt1 = zeros(2,its); % Fourier coef. -- Std dev. = 1.0

% MOVE THE IMAGE ACROSS THE DIGITAL ARRAY FOR THE NUMBER OF
% ITERATIONS SPECIFIED.

for i = 0:(its-1),

% DEFINE THE PARAMETERS FOR IMAGE LOCATION

    i1 = rem(round(vrow*i+1),32)+1;
    i2 = rem(round(vrow*i+8),32)+1;
    i3 = rem(round(vclm*i+1),32)+1;
    i4 = rem(round(vclm*i+8),32)+1;
```

```

% PUT IMAGE INTO THE 32X32 ARRAY "pict"

pict = zeros(32);

if i2 < i1,
    if i4 < i3,
        pict(i1:32,i3:32) = shape(1:n1-i2,1:n2-i4);
        pict(1:i2,1:i4)    = shape(n1+1-i2:n1,n2+1-i4:n2);
        pict(i1:32,1:i4)   = shape(1:n1-i2,n2+1-i4:n2);
        pict(1:i2,i3:32)   = shape(n1+1-i2:n1,1:n2-i4);
    else
        pict(i1:32,i3:i4) = shape(1:n1-i2,1:n2);
        pict(1:i2,i3:i4)  = shape(n1+1-i2:8,1:n2);
    end
else
    if i4 < i3,
        pict(i1:i2,i3:32) = shape(1:n1,1:n2-i4);
        pict(i1:i2,1:i4)  = shape(1:n1,n2+1-i4:n2);
    else
        pict(i1:i2,i3:i4) = shape(1:n1,1:n2);
    end
end

% CREATE THE FRAME CORRUPTED BY NOISE

pict1 = pict + (sig1*rand(32));

% TRANSFORM THE FRAMES INTO THE SPATIAL FREQUENCY DOMAIN

fftpict0 = fftshift(fft2(pict));
fftpict1 = fftshift(fft2(pict1));

% OBTAIN THE PHASE IN DEGREES FROM SPATIAL FREQUENCIES
% F(0,1) & F(1,0)

phase0(1,i+1) = 57.2958*angle(fftpict0(16,17));
phase1(1,i+1) = 57.2958*angle(fftpict1(16,17));
phase0(2,i+1) = 57.2958*angle(fftpict0(17,18));
phase1(2,i+1) = 57.2958*angle(fftpict1(17,18));

% OBTAIN THE FOURIER COEFFICIENTS FOR SPATIAL FREQUENCIES
% F(0,1) & F(1,0)

rslt0(1,i+1) = fftpict0(16,17);
rslt1(1,i+1) = fftpict1(16,17);
rslt0(2,i+1) = fftpict0(17,18);
rslt1(2,i+1) = fftpict1(17,18);

end

```

```
% PLOT THE PHASE SHIFT FOR SPATIAL FREQ F(0,1) IN DEGREES
```

```
n = 1:its;
plot(n,phase0(1,:), 'w', n, phase1(1,:), '--w')
title('Phase shift for image velocity of (1,2)')
xlabel('Phase shift for F(0,1) in degrees')
ylabel('Degrees')
meta pphse
pause
```

```
% PLOT THE PHASE SHIFT FOR SPATIAL FREQUENCY F(1,0) IN
% DEGREES
```

```
plot(n,phase0(2,:), 'w', n, phase1(2,:), '--w')
title('Phase shift for image velocity of (1,2)')
xlabel('Phase shift for F(1,0) in degrees')
ylabel('Degrees')
meta pphse
pause
```

```
% PLOT THE REAL FOURIER COEFF FOR SPATIAL FREQ F(0,1)
```

```
plot(n,real(rslt0(1,:)), 'w', n, real(rslt1(1,:)), '--w')
title('Real Fourier coefficients for F(0,1)')
xlabel('Iteration'), ylabel('Magnitude')
meta pphse
pause
```

```
% PLOT THE IMAGINARY FOURIER COEFF FOR SPATIAL FREQ F(0,1)
```

```
plot(n,imag(rslt0(1,:)), 'w', n, imag(rslt1(1,:)), '--w')
title('Imaginary Fourier coefficients for F(0,1)')
xlabel('Iteration'), ylabel('Magnitude')
meta pphse
pause
```

```
% PLOT THE REAL FOURIER COEFF FOR SPATIAL FREQ F(1,0)
```

```
plot(n,real(rslt0(2,:)), 'w', n, real(rslt1(2,:)), '--w')
title('Real Fourier coefficients for F(1,0)')
xlabel('Iteration'), ylabel('Magnitude')
meta pphse
pause
```

```
% PLOT THE IMAGINARY FOURIER COEFF FOR SPATIAL FREQ F(1,0)
```

```
plot(n,imag(rslt0(2,:)), 'w', n, imag(rslt1(2,:)), '--w')
title('Imaginary Fourier coefficients for F(1,0)')
xlabel('Iteration'), ylabel('Magnitude')
meta pphse
pause
```


APPENDIX C. PMEKF.M

PROGRAM LISTING

PMEKF.M is a stand-alone MATLAB program that implements the modified extended Kalman filter (MEKF) for moving objects.

```
% *****
% VARIABLE DEFINITIONS
%
% shpe      : string defining shape of moving object
% vrow      : vertical velocity in # of pixels per iteration
% vclm      : horiz. velocity in # of pixels per iteration
% imax,jmax : rectangular dimensions of spatial truncation
%            window; located at center of transformed frame
% its       : number of iterations
% sig       : standard deviation of the added noise
% chkrbd    : flag indicating user's intentions of whether
%            or not to include checkerboard background
% a         : amplitude of individual checkers
% runnum    : for successive runs; number indicating
%            specific run
% pictchkr  : image frame containing checkerboard bkgrd;
%            moving image included later
% freq      : vector listing two-dimensional frequencies to
%            be analyzed
% lfreq     : length of the freq vector
% normpic   : Frobenius norm of error matrix; subtract
%            filtered image frame from original, no-noise
%            image frame
% normpict  : Frobenius norm of error matrix; subtract
%            filtered image frame from truncated, no-noise
%            image frame
% xh        : state estimate vector
% vh        : velocity estimate vector
% y         : vector containing Fourier coefficients
% pk        : storage array for "pkk" covariance matrices
% pkk       : estimation error covariance matrix; spatial
%            frequency specific
% q         : plant noise covariance matrix
% r         : measurement noise covariance matrix
% pict      : no-noise image frame containing moving object
% pictn     : no-noise image frame containing moving object
%            and spatial frequency truncated
% randpict  : image frame containing moving object and added
%            zero-mean, white Gaussian noise
% *****
```

```

clear
clear functions

    shpe = ['Rectangle'];
    vrow = 2.0;
    vclm = 1.0;
    imax = 3;
    jmax = 3;
    its = 25;
    sig = 1.00;
    chkrbd = 'n';          % ( 'y' -- yes; 'n' -- no )
    a = 0.25;
    runnum = 1;

% DEFINE THE SHAPE

x1 = 1.00; x2 = 1.00; x3 = 1.00; x4 = 1.00;

shape = [ [ x1 x1 x1 x1 x1 x1 x1 x1 ]
          [ x1 x2 x2 x2 x2 x2 x2 x1 ]
          [ x1 x2 x3 x3 x3 x3 x2 x1 ]
          [ x1 x2 x3 x4 x4 x3 x2 x1 ]
          [ x1 x2 x3 x4 x4 x3 x2 x1 ]
          [ x1 x2 x3 x3 x3 x3 x2 x1 ]
          [ x1 x2 x2 x2 x2 x2 x2 x1 ]
          [ x1 x1 x1 x1 x1 x1 x1 x1 ] ];

% CREATE A CHECKERBOARD BACKGROUND IF REQUIRED

if (chkrbd == 'y'),
    pictchkr = zeros(32);
    chkr = a * ones(8);
    for i = 1:4,
        if i == 1 | i == 3,
            pictchkr([i*8-7:i*8],[1:8]) = chkr;
            pictchkr([i*8-7:i*8],[17:24]) = chkr;
        else
            pictchkr([i*8-7:i*8],[9:16]) = chkr;
            pictchkr([i*8-7:i*8],[25:32]) = chkr;
        end
    end
end
end

```

```

% CREATE FREQUENCY VECTOR

n = 0;
for i = 0:imax,
    if i == 0,
        for j = 0:jmax,
            n = n+1;
            freq(n,:) = [ i j ];
        end
    else
        for j = -jmax:jmax,
            n = n+1;
            freq(n,:) = [ i j ];
        end
    end
end

% CREATE AND DEFINE NECESSARY VARIABLES

co = pi/16;
jj = sqrt(-1);
lfreq = length(freq);
rand('normal'); % Establish normal distribution
rand('seed',2380849) % Define a standard random seed
[ n1 n2 ] = size(shape); % Dimensions of the image frame
y = zeros(lfreq,its);
xh = zeros(3,lfreq);
vh = zeros(2,its);
q = [ 0 0 0
      0 0 0
      0 0 .4 ];
r = sig^2*512*eye(2);

normpic = zeros(1,its);
normpict= zeros(1,its);

% GENERATE THE INITIAL ESTIMATION ERROR COVARIANCE MATRIX.
% EACH INDIVIDUAL FREQ COVARIANCE MATRIX IS STACKED IN "pk"

pk = zeros(9,lfreq);
for k = 1:lfreq,
    pkk = [ 1024 0 0
            0 1024 0
            0 0 4*freq(k,:)*freq(k,:)'];
    pk(:,k) = pkk(:);
end

```

```

% MODIFIED EXTENDED KALMAN FILTER

% OBJECT IS MOVED ACROSS THE IMAGE FRAME FOR THE NUMBER OF
% ITERATIONS SPECIFIED

for i = 0:(its-1),

% REINITIALIZE THE TYPE OF IMAGE FRAME USED IN THE
% SIMULATION

    if (chkrbd == 'y'),
        pict = pictchkr;
    else
        pict = zeros(32);
    end

% DEFINE THE PARAMETERS FOR IMAGE LOCATION

    i1 = rem(round(vrow*i+1),32)+1;
    i2 = rem(round(vrow*i+8),32)+1;
    i3 = rem(round(vclm*i+1),32)+1;
    i4 = rem(round(vclm*i+8),32)+1;

% PUT IMAGE INTO THE 32X32 ARRAY "pict"

    if i2 < i1,
        if i4 < i3,
            pict(i1:32,i3:32) = shape(1:n1-i2,1:n2-i4);
            pict(1:i2,1:i4)    = shape(n1+1-i2:n1,n2+1-i4:n2);
            pict(i1:32,1:i4)   = shape(1:n1-i2,n2+1-i4:n2);
            pict(1:i2,i3:32)   = shape(n1+1-i2:n1,1:n2-i4);
        else
            pict(i1:32,i3:i4) = shape(1:n1-i2,1:n2);
            pict(1:i2,i3:i4)  = shape(n1+1-i2:8,1:n2);
        end
    else
        if i4 < i3,
            pict(i1:i2,i3:32) = shape(1:n1,1:n2-i4);
            pict(i1:i2,1:i4)   = shape(1:n1,n2+1-i4:n2);
        else
            pict(i1:i2,i3:i4) = shape(1:n1,1:n2);
        end
    end

    randpict = pict + (sig*rand(32));
    fpic     = fft2(randpict);
    fpic     = fftshift(fpic);
    fband    = fpic((17-imax):(17+imax),(17-jmax):(17+jmax));
    fvec     = fband(:);
    y(:,i+1) = fvec(((length(fvec)+1)/2):length(fvec));

```

```
% TRUNCATE THE SPATIAL FREQS OF UNCORRUPTED PICTURE
```

```
pictn    = fft2(pict);
pictn    = fftshift(pictn);
fband1   =pictn((17-imax):(17+imax),(17-jmax):(17+jmax));
pictn    = zeros(32);
pictn((17-imax):(17+imax),(17-jmax):(17+jmax)) = fband1;
pictn    = fftshift(pictn);
pictn    = ifft2(pictn);
```

```
% ITERATE THRU THE "parallel" BANK OF EXT. KALMAN FILTERS
```

```
for k = 1:lfreq,
    dth = co*xh(3,k);
    cdth = cos(dth);
    sdth = sin(dth);
    dh = [ cdth sdth
           -sdth cdth ];
    dph = [ (-xh(1,k)*sdth+xh(2,k)*cdth)*co
            (-xh(1,k)*cdth-xh(2,k)*sdth)*co ];

    ah = [ dh dph
           0 0 1 ];
    pkk(:) = pk(:,k);
    pkp1k = ah*pkk*ah' +q;
    g = pkp1k(:,1:2)/(pkp1k(1:2,1:2)+r);
    xh(1:2,k) = dh*xh(1:2,k);
    yh = xh(1:2,k);
    ym = [ real(y(k,i+1))
           imag(y(k,i+1)) ];
    xh(:,k) = xh(:,k)+g*(ym-yh);
    pkk = pkp1k - g*[1 0 0; 0 1 0]*pkp1k;
    pk(:,k) = pkk(:);
end
```

```
% THESE THREE LINES GENERATE THE WEIGHTED LEAST SQUARES
% ESTIMATE OF THE VELOCITY VECTOR. COMMENTED OUT WHEN
% NOT USED.
```

```
sigin = diag(ones(1,(lfreq-1)) ./pk(9,2:lfreq));
fre = freq(2:lfreq,:);
vh(:,i+1) = (fre'*sigin*fre)\fre'*sigin*xh(3,2:lfreq)';
```

```
% THIS LINE GENERATES THE LEAST SQUARES ESTIMATE OF THE
% VELOCITY VECTOR (unweighted). COMMENTED OUT WHEN NOT
% USED.
```

```
% vh(:,i+1) = freq(2:lfreq,:)\xh(3,2:lfreq)';
```

```
% THIS LINE CHANGES THE ESTIMATES OF FREQUENCY FOR THE
% INDIVIDUAL FILTERS LEAST SQUARES ESTIMATE.  COMMENTED OUT
% WHEN NOT USED.
```

```
xh(3,:) = [ freq*vh(:,i+1) ]';
```

```
% RECONSTRUCT THE IMAGE BACK INTO THE TIME DOMAIN.
```

```
fpic = zeros(32);
xhc = zeros(1,(2*lfreq-1));
xhc(lfreq:(lfreq*2-1)) = xh(1,:)+jj*xh(2,:);
xhc(lfreq:-1:1) = xh(1,:)-jj*xh(2,:);
xhc(lfreq) = xh(1,1);
fband(:) = xhc;
fpic((17-imax):(17+imax),(17-jmax):(17+jmax)) = fband;
fpic = fftshift(fpic);
pic = ifft2(fpic);
```

```
% CALCULATE THE NORM OF A MATRIX
```

```
% FROBENIUS NORM : UNCORRUPTED IMAGE
```

```
normpic(i+1) = norm(pict - pic,'fro')/ 32;
```

```
% FROBENIUS NORM : UNCORRUPTED, SPATIALLY-TRUNCATED IMAGE
```

```
normpict(i+1)= norm(pictn - pic,'fro')/ 32;
```

```
end
```

```
% DEFINE A TEXT LABEL FOR FINAL PLOTS
```

```
name = ['Std deviation = ',num2str(sig),' -- '...
        'Iterations = ',num2str(its),' -- '...
        'vel(',num2str(vrow),',',num2str(vclm),')'];
```

```
% DISPLAY THE FINAL IMAGE
```

```
pname = ['meta pmekf_',num2str(runnum) ];
```

```
clc
mesh(pic)
title([ shpe ' -- Filtered Image -- Time domain'])
xlabel(name)
eval(pname)
pause
```

```

mesh(pict)
title([ shpe ' -- Uncorrupted image'])
xlabel(name)
eval(pname)
pause

mesh(pictn)
title([ shpe ' -- Uncorrupted image (spatially truncated)'])
xlabel(name)
eval(pname)
pause

% PLOT FROBENIUS NORMS AND VELOCITY ESTIMATES

n = 1:its;
plot(n,normpic,'-',n,normpict,'--')
title([ shpe ' -- Frobenius Norm'])
eval(pname)
pause

plot(n,vh(1,:),'-',n,vh(2,:),'--')
title([ shpe ' -- Velocity Estimates'])
xlabel(name)
eval(pname)
pause

% SAVE THE CALCULATED DATA

m = [ n' vh' normpic' normpict'];
ekfdata=['save pmekf_',num2str(runnum),'.mat m /ascii'];
eval(ekfdata);

```

LIST OF REFERENCES

1. Burl, J.B., *A Reduced Order Extended Kalman Filter for Moving Images*, unpublished manuscript, Naval Postgraduate School, Monterey, California, 1989.
2. Dudgeon, Dan E. and Mersereau, Russell M., *Multidimensional Digital Signal Processing*, pp. 61-67, Prentice-Hall, 1984.
3. Brigham, E. Oran, *The Fast Fourier Transform*, p. 45, Prentice-Hall, 1974.
4. Candy, James, *Signal Processing: The Model-Based Approach*, pp. 154-159, McGraw-Hill Book Company, 1986.
5. Anderson, Brian D.O. and Moore, John B., *Optimal Filtering*, pp. 193-195, Prentice-Hall, 1979.
6. Gelb, A. et al, *Applied Optimal Estimation*, pp. 278-280, The M.I.T. Press, 1986.
7. Golub, Gene H. and VanLoan, Charles F., *Matrix Computations*, The John Hopkins University Press, 1983.

BIBLIOGRAPHY

1. Friedland, Bernard, *Control System Design: An Introduction to State-Space Methods*, McGraw Hill Book Company, 1986.
2. Gonzalez, Rafael C and Wintz, Paul, *Digital Image Processing*, 2nd ed., Addison-Wesley Publishing Company, 1987.
3. Kirk, D.E., *Optimal Estimation: An Introduction to the Theory and Applications*, unpublished manuscript, Naval Postgraduate School, Monterey, California, 1975.
4. Strang, Gilbert, *Linear Algebra and Its Applications*, 3rd ed., Harcourt Brace Jovanovich Publishers, 1988.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
4. Profesosr R. Cristi, Code 62Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
5. Professor J.B. Burl, Code 62B1 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5004	1
6. Commandant of the Marine Corps Code TE 06 Headquarters, U.S. Marine Corps Washington, DC 20380-0001	1
7. Captain P.A. Lindeman, U.S.M.C. 220 68 8708/7654/9624 MCRADC Marine Corps Combat Development Center Quantico, Virginia 22134-5080	2
8. Lieutenant W.A. Conklin, USN Code 39 Naval Postgraduate School Monterey, California 93943-5000	1

- | | | |
|-----|----------------------------------|---|
| 9. | Captain S.L. Spehn, U.S.M.C. | 1 |
| | Code 32 | |
| | Naval Postgraduate School | |
| | Monterey, California 93943-5000 | |
| 10. | Major D. Aviv, Israeli Air Force | 1 |
| | Code 32 | |
| | Naval Postgraduate School | |
| | Monterey, California 93943-5000 | |